

早わかり
マシン語事典

春田正夫／沢田 昭

MACHINE

LANGUAGE

新星出版社

早わかり マシン語事典

春田正夫／沢田 昭



新星出版社

はじめに

かつて、ワンボード・マイコンがありました。そのときのプログラミング言語はマシン語でした。そのあとパソコンになって、プログラミング言語はベーシックになりました。そしていままた、プログラミング言語はマシン語、と思えるほどマシン語が盛んになってきましたが、これからはベーシックとかマシン語とかの一方に偏らずに、ベーシックの長所とマシン語の長所を組み合わせるプログラムを作っていくほうが簡単で、しかも、面白いプログラムを作ることができると思います。

そのためには、マシン語を勉強しなければなりません。ベーシックと違って、マシン語についてはマニュアルを見ても解説されていません。解説されているのは、マシン語のコマンドだけです。これでは、マシン語を学んでみようと勉強することができません。そこで、これからマシン語を学んでみようという人のために、この本を書きました。

この本は、これからマシン語を勉強しようという人のためのものですから、まず、マシン語を使うとき知っておくと役に立つ事柄について説明しました。

つぎにマシン語の命令のなかから、基礎的な命令を選び出して、だれでもわかるように詳しく説明しました。説明の仕方は、単にマシン語の命令を取りあげて、その命令の働きを説明しても意味がないので、キャラクタを画面に表示させたり、移動させたりして、そのプログラムのなかでマシン語の命令やその働きが理解できるようにしました。このようにすると、マシン語の命令の働きばかりでなく、マシン語のプログラムがどのように作られているかということも、理解することができると思ったからです。

この本で取りあげたマシン語は、Z-80CPUのマシン語です。したがってZ-80CPUを搭載しているパソコンなら、どの機種にも共通しています。ただマシン語が共通であっても、機種によって構造が異なります。機種によって異なる点は、マシン語を記憶するメモリ領域と、キャラクタを表示するVRAMのメモリ番地です。

この本を書くにあたって使った機種は、PC-8001です。したがって、PC-8001mkII、PC-8801およびMKIIのN-BASICモードでは、この本をそのまま活用することができます。そのほかの機種は、さきに述べた点に注意してこの本を活用してください。

も く じ

マシン語が使える状態にする	
ベーシックが使える状態に戻す	12
MONコマンドで、マシン語が使える状態にする(12)	
ベーシックのコマンド・レベルに戻すには(14)	
マシン語のプログラムを入力するには	15
マシン語のプログラムを入力するには、Sコマンド(15)	
Sコマンドは、セット・メモリのこと(16)	
Sコマンドの入力を忘れると(20)	
メモリに記憶させたマシン語のプログラムを 画面に表示させる	21
Dコマンドは、ダンプ・メモリの意味(21)	
16個以上のメモリの内容を表示させるには(23)	
表示を止める、表示を再開する、表示を中止する(24)	
マシン語のプログラムを実行させる	26
マシン語のプログラムの実行はGコマンド(26)	
GコマンドはG Oのこと(27)	
ふたたび実行させる(29)	
入力したマシン語プログラムを変更する方法	31
変更するときは、Sコマンドを使う(31)	
変更する必要のないところはスペースキーで飛ばす(34)	
まえのメモリ番地に戻すときは(36)	
間違ってマシン語(16進数)にないものを入力したとき(38)	

コマンドのあとのメモリ番地の入力を間違えたときは40

メモリ番地の入力を間違えたときは **CTRL** + **B** キー (40)

マシン語のプログラムをカセットテープに記録する44

プログラムをカセットテープに記録するには **W** コマンド (44)

W コマンドのあとにはメモリ番地を置く (44)

ベリファイを行うには47

ベリファイを行うには、**LV** コマンド (47)

ベリファイを行うには、**LV** ☒ (47)

カセットテープに記録したプログラムを メモリのなかに読み込むには50

メモリのなかに読み込むには、**L** コマンド (50)

読み込まれたプログラムが記憶されるメモリ番地は ? (52)

ベーシックとマシン語でできたプログラムを カセットテープに記録する53

ベーシックを記録するには **CSAVE**、マシン語は **W** (53)

まず、ベーシックのプログラムを記録する (55)

つぎに、マシン語のプログラムを記録する (55)

ベリファイを行うには (56)

ベーシックとマシン語でできたプログラムを パソコンのメモリに読み込む60

ベーシックを読み込むには **CLOAD**、マシン語は **L** (60)

まず、ベーシックのプログラムを読み込む (61)

つぎに、マシン語のプログラムを読み込む (61)

パソコンのメモリの良否をチェックするには63

メモリのチェックは、**TM** コマンド (63)

メモリ番地の読み方65

8 バイト表示のとき、メモリ番地は 8 つ置きに表示される (65)

チェックサムとは68

チェックサムは、入力ミスがあるかないかを調べる (68)

チェックサムのつけ方 (69)

入力したマシン語のチェックサムを求める (72)

チェックサム・プログラムの入力 (72)

ビットとバイト79

1 ビットは 2 進数の 1 桁を表わす (79)

8 ビットは 1 バイト (80)

ひとつのメモリも 8 ビットを記憶する (80)

16 進数 1 桁は 4 ビット (80)

16 進数の 2 桁は 1 バイト (80)

2 進数は、このように書き表わされている81

2 進数は、1、2、4、8 の値に電気をつけて数を表わす (81)

オンに 1、オフに 0 をあてはめる (83)

8 ビットの場合は (84)

10 進数と 8 ビットの 2 進数対応表 (86)

マシン語は 16 進数で書き表わす91

16 進数は、16 で桁上がりする数 (91)

4 ビットの 2 進数は、16 進数の 1 桁で表わされる (93)

8 ビットの 2 進数は、16 進数 2 桁で表わされる (94)

メモリには 1 バイトずつ記憶する (95)

2 進数を 16 進数へ、16 進数を 2 進数へ変換するには97

2 進数 \longleftrightarrow 16 進数変換は 4 ビットずつで行う (97)

2 進数を 16 進数へ変換する (97) / 16 進数を 2 進数へ変換する (100)

2進数を10進数へ、10進数を2進数へ変換するには …… 102

ビット番号にふられた値に基づいて変換する (102)

2進数を10進数に変換する (102) / 10進数を2進数に変換する (104)

マシン語のプログラムを書くには …… 107

アセンブリ言語、アセンブル、アセンブラ (107)

まず、アセンブラでプログラムを書く (107)

アセンブルは、アセンブリ言語をマシン語になおすこと (108)

アセンブルには、Z-80アセンブル表が必要 (108)

画面への表示 …… 109

LDは、ベーシックのLETとおなじ働き (109)

LDは、ロード命令という (109)

Aは、Aレジスタ (110)

LD (F300H), Aとは (111)

HALTは、END (112)

アセンブリ言語をマシン語になおす (113)

メモリ上に割りつける (115)

あるメモリ番地からあるメモリ番地へ ジャンプさせるには …… 116

JPは、ベーシックのGOTO (116)

JP命令は、無条件ジャンプ (116)

JP命令は、マシン語モニタに戻す (117)

アセンブリ言語をマシン語に直す (118)

メモリ上に割りつける (119)

ベーシックのコマンド・レベルに戻すには (120)

ベーシックのプログラムをも消滅させない方法 (121)

画面の左から右へ複数個のキャラクタを表示する …… 123

基本的な加算命令 INC (123)

INC命令は、+ 1 する命令 (125)
INC命令を使って、複数個のキャラクタを表示する (126)
HLレジスタは、HとLレジスタを組み合わせたもの (127)
アセンブリ言語をマシン語になおす (129)
横40文字表示で、ハートを 5 個表示するには (131)

画面の右から左へ複数個のキャラクタを表示する 132

基本的な減算命令 DEC (132)
DEC命令は、- 1 する命令 (132)
DEC命令を使って、複数個のキャラクタを表示する (134)
アセンブリ言語をマシン語になおす (136)
横40文字表示で、○を 5 個表示するには (137)
引く数だけDEC HL命令を重ねる (138)

ベーシックのFOR~NEXT命令とおなじ繰り返えし処理をさせるには 139

繰り返えし処理専門の命令 DJNZ (139)
繰り返えし処理をする回数はBレジスタにセット (140)
40個の●を画面に表示する (141)
アセンブリ言語をマシン語になおす (144)
10のあとのe- 2 とは (144)
相対アドレス指定のやり方 (145)

画面の上から下に向けて●を表示 148

16ビット (2 バイト) の加算命令 ADD (148)
●を画面の下に向けて表示する (149)
ADD命令で、メモリ番地に120を加算 (150)
DJNZ命令で戻るべきメモリ番地を指定 (152)
HALTはENDとおなじ (154)

画面の下から上に向けて●を表示 155

16ビット (2 バイト) の減算命令 SBC (155)

- を画面の下から上に向けて表示する (156)
- SBC命令で、HLレジスタから120を引く (158)
- DJNZ命令で戻るメモリ番地の指定 (159)

画面設定の仕方 162

- CALL命令でモニタサブルーチンを呼ぶ (162)
- BレジスタとCレジスタに値をセット (163)

画面の下から上に向けてキャラクタを移動する 165

- まえに表示したキャラクタを消すには XOR A (165)
- XOR Aで、空白のキャラクタコードを作る (167)
- を画面の下から上へ移動する (168)
- DJNZ命令で戻すところはLD A, ECH (168)
- XOR Aで、OOHを算出する (169)
- SBC命令で、つぎに表示するメモリ番地を算出 (169)
- DJNZ命令で、繰り返えし処理するために戻す (170)

画面の上から下に向けてキャラクタを移動する 172

- ポインタはXOR Aと、ADD HL, X (172)
- 画面設定をする (173)
- VRAMのメモリ番地などを決める (173)
- キャラクタの表示と消去 (175)
- キャラクタを表示するメモリ番地の計算 (175)
- DJNZ命令で戻るメモリ番地の指定 (176)

命令の実行を遅らせるには 178

- ウェイト・ループ (178)
- フラグとは (179)
- JP NZ命令は、ゼロ・フラグが0のときだけジャンプ (180)
- キャラクタを画面の下から上へ移動する (181)
- それぞれの命令の働き (182)
- PUSH命令で、Bレジスタの内容を退避させる (183)

ウェイト・ループを入れる (184)
POP 命令で退避させた B レジスタの内容を取り出す (184)
XOR A 命令で、空白のキャラクタコードを作る (185)
マシン語のプログラム (186)
キャラクタを画面の上から下へ移動する (187)

画面に表示されているものをすべてを消す 188

比較命令 CP を使って (188)
VRAM のメモリ番地を指定して、00H を表示する (189)
HL レジスタの数値の記憶の仕方 (190)
CP 命令で、上位ビットが FEH になったかを比較 (190)
JP NZ 命令で判定して、その後の行動をする (191)
CP 命令で、下位ビットが B7H になったかを比較 (191)

UFO やタンクなどのキャラクタを作るには 193

グラフィック・シンボルをキャラクタコードになおす (193)

タンクなどのキャラクタを画面に表示する 195

キャラクタのデータは、LD A, (DE) で読み取る (195)
データの最初のメモリ番地をレジスタに代入 (196)
データを読み取らせるには (196)
データの読み取りと、表示を繰り返す (197)
カウントする命令を置く (198)
PUSH 命令で H レジスタ、B レジスタの内容を退避 (201)
2 行目、3 行目にキャラクタを表示するメモリ番地の計算 (202)
DJNZ 命令で、繰り返し処理をする最初の命令に戻る (203)
アセンブルしたプログラム (204)

タンクから弾を発射する 206

キー入力、IN 命令 (206)
CP 命令で比較する (207)
JP Z 命令は、ゼロ・フラグが 1 のときジャンプする (208)
弾を発射するタンクを表示する (209)

- 弾丸を発射する (209)
- タンクを表示するデータ (213)
- アセンブルしたプログラム (214)

UFOを画面の左から右に飛ばす 216

- CP命令で比較、JP NZ命令で判定して繰り返す (216)
- UFOを画面の右へ向けて移動する (219)
- スタックへのデータの記憶のされ方 (221)
- UFOのデータを読み取って表示 (222)
- つぎのキャラクタの表示のために、INC命令で1プラス (223)
- ウェイト・ループを入れる (228)
- UFOを表示するつぎのメモリ番地の指定 (228)
- アセンブルしたプログラム (231)

UFOを画面の右から左へ飛ばす 233

- INC HL命令を、DEC HL命令に変える (233)
- アセンブルしたプログラム (234)

画面の左右にきたUFOの消去 236

- 00HをUFOの上に表示する (236)
- 00Hを表示する最初のVRAMのメモリ番地を求める (237)
- 00Hの表示 (238)
- アセンブルしたプログラム (240)

タイトルの表示 243

〈付 録〉

- 2進 \longleftrightarrow 16進変換表 (245)/レジスタの種類 (245)
- Z-80アセンブル表 (246)/10 \longleftrightarrow 16進変換表 (248)
- 1バイト符号付16進数 (249)/VRAMメモリ番地表 (250)
- キャラクタ・コード表 (254)

マシン語が使える状態にする
ベーシックが使える状態に戻す

MONコマンドで、マシン語が使える状態にする

ふつうパソコンは、ふたつの状態（コマンド・レベル）を持っています。ひとつは、ベーシックが使える状態（ベーシックのコマンド・レベル）、もうひとつは、マシン語が使える状態（マシン語のコマンド・レベル）です。

パソコンに電源を入れると、つぎに示すような状態になります。

実行結果

Ok ← ベーシックのコマンド・レベル (ベーシックが使える状態)

——カーソル

auto go to list run

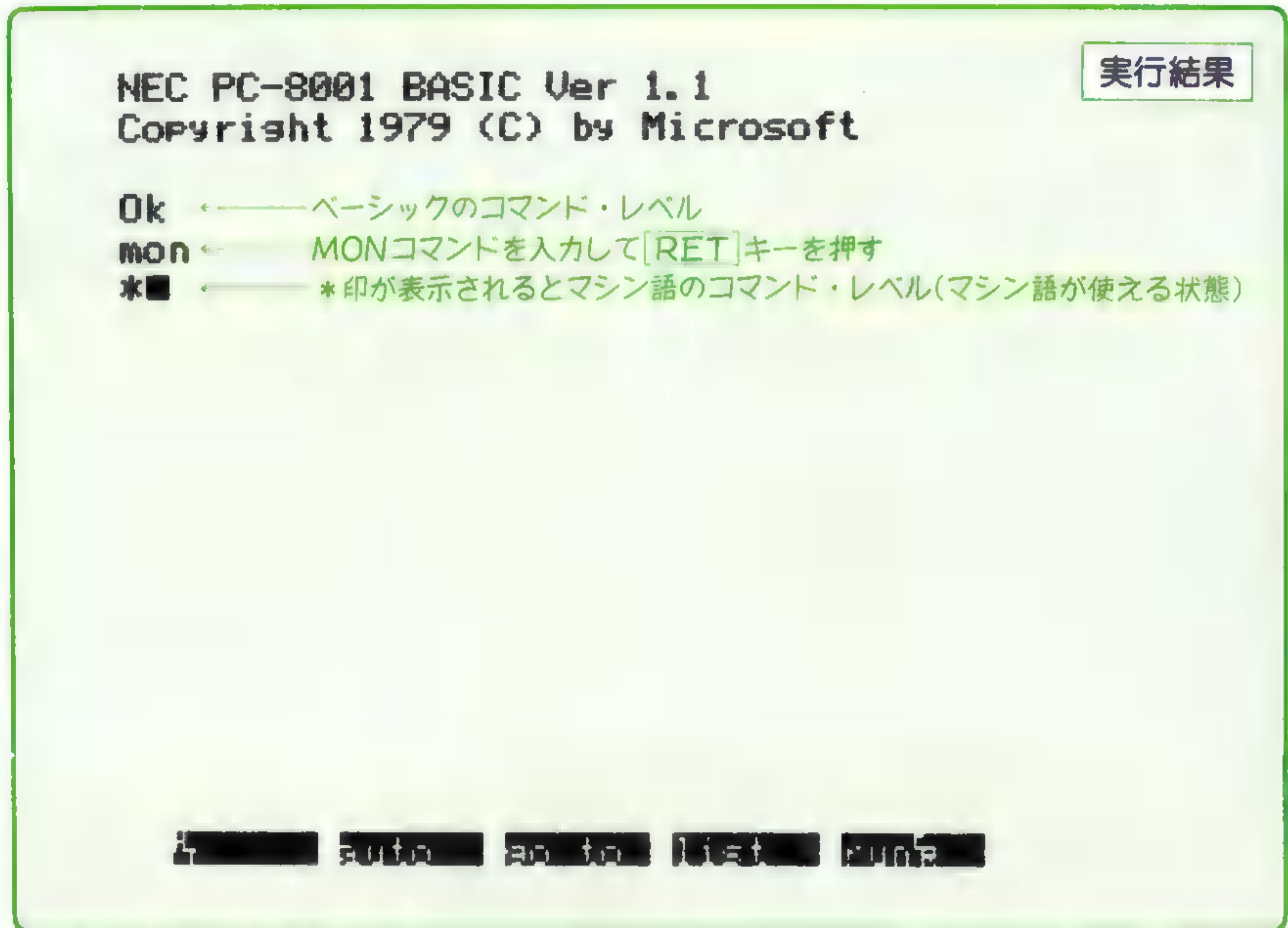
画面にOKが表示され、カーソルが点滅している状態は、ベーシックのコマンド・レベルで、ベーシックのプログラムを入力することはできますが、マシン語のプログラムを入力することはできません。


今日のパソコンは、だれもがパソコンを使うことができるように、**スイッチ・オン・ベーシック**になっています。そのため、電源を入れると、まずベーシックのコマンド・レベルになるわけです。

マシン語が使える状態にする

したがって、マシン語が使える状態つまりマシン語のコマンド・レベルにするには、**MONコマンド**を入力して、ベーシックのコマンド・レベルを、マシン語のコマンド・レベルに切り換える必要があります。

MON



上の図に示すように、MONコマンドを入力して**RET**キーを叩きます。なお、の印は、**RET**キーを叩くことを意味しています。

mon  ← マシン語のコマンド・レベルにする

すると、MONコマンドの下に*印（アスタリスク）が表示されます。

mon

* ← マシン語のコマンド・レベルの印

この*印が表示された状態がマシン語のコマンド・レベルです。これでマシン語のプログラムを入力したり、入力したプログラムを実行させたりすることができるようになります。

OK ← ベーシックのコマンド・レベル

* ← マシン語のコマンド・レベル

ベーシックのコマンド・レベルに戻すには

では、マシン語のコマンド・レベルになったところで、ふたたびベーシックのコマンド・レベルに戻すことにします。マシン語のコマンド・レベルからベーシックのコマンド・レベルに戻す方法を知らないと、パソコンのうしろにあるリセットボタンを押したり、リセットボタンを押すということを知らないときは、パソコンの電源を切ったりしなければならなくなります。

マシン語のコマンド・レベルからベーシックのコマンド・レベルに戻すには **CTRL** キーとアルファベットの **B** のキーを一緒に押します。

CTRL + **B**

この場合、**RET** キーを叩く必要はありません。ふたつのキーを一緒に押すだけです。すると、カーソルが点滅していた位置に、つぎの図に示すような記号を表示して、その下に **OK** を表示します。

OK が表示された状態は、ベーシックのコマンド・レベルですから、これでマシン語のコマンド・レベルからベーシックのコマンド・レベルに戻ったことになります。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok ← ベーシックのコマンド・レベル、
mon ← マシン語のコマンド・レベルにする
*5 ← **CTRL** キーと **B** のキーを一緒に押す
Ok ← ベーシックのコマンド・レベルに戻る
■

h auto auto list func

マシン語のプログラムを入力するには

マシン語のプログラムを入力するには、Sコマンド

12頁でマシン語のコマンド・レベルに切り換える方法を説明しました。ここでは、マシン語のプログラムの入力の仕方について説明します。

つぎに示すのがマシン語のプログラムです。プログラムについては 107 頁から説明するので、ここでは説明を省略しますが、マシン語のプログラムはつぎのように、16進数で書き表わす、ということだけ覚えておいてください。

```
3E E9
32 E4 F6
76
```

マシン語のプログラム

このマシン語のプログラムをベーシックで書くと、つぎのようになります。

```
10 LOCATE 18, 8
20 PRINT "♥"
30 END
```

マシン語のプログラムを入力して記憶させる場合は、メモリマップを見て、ユーザーエリアと書かれた部分を使いますから、上に示したマシン語のプログラムを入力するには、どの番地のメモリから入力して記憶させるか、まず決めなければなりません。右に示したメモリマップは、PC-8001MKII(32K)のメモリマップです。PC-8001MKII(32K)の場合は、8021番地からE9FF番地までがユーザーエリアとなっていますから、8021番地からE9FF番地まで使うことができますようになっています。



メモリマップ

ここでは、D000番地のメモリからマシン語のプログラムを入力することにします。そこでさきのマシン語のプログラムに入力して記憶させるメモリ番地をふると、つぎのようになります。

メモリ番地	マシン語	メモリ番地	マシン語
D000	3E	D003	E4
D001	E9	D004	F6
D002	32	D005	76

D000番地のメモリに3E、D001番地のメモリにE9、D002番地のメモリに32というように、ひとつのメモリに、16進数の2桁を入力して記憶させます。その理由は、ひとつのメモリには1バイト（8ビット）しか記憶させることができないからです。

16進数の場合は、2桁で1バイトです。したがってさきのマシン語のプログラムは、D000番地のメモリからD005番地までに入力して記憶させることになります。

Sコマンドはセット・メモリのこと

では、マシン語のプログラムの入力です。まず、MONコマンドを入力して、マシン語のコマンド・レベルにします。

mon 

*  マシン語のコマンド・レベルにする

*印が表示されたら、Sコマンドを入力します。Sコマンドは、SET MEMORY（セット・メモリ）のことで、マシン語のプログラムを入力するときに使います。

S〈プログラムを入力する最初のメモリ番地〉

マシン語のプログラムを入力するときは、Sコマンドのあとに、マシン語のプログラムを入力する最初のメモリ番地を入力します。この場合は、D000番地からマシン語のプログラムを入力していきますから、SコマンドのあとにD000番地を入力することになります。

mon

* S D000   SコマンドのあとにD000番地を入力する

マシン語のプログラムを入力するには

D000番地を入力したら、**RET** キーを叩きます。するとつぎの実行結果に示すように、最初のメモリ番地D000が表示されます。そして、D000番地のあとにFF-（機種によっては、00-）が表示されます。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok

mon

*SD000

D000 FF-

← マシン語のコマンド・レベルにする

← Sを入力して、最初のメモリ番地D000を入力する

← D000番地とFF-（または00-）が表示される

Auto Run

一の左側にある最初のFFは、D000番地に記憶されているデータです。メモリのなかになにもデータが入力されていない場合は、FFか00がデータとして記憶されています。したがって、それを表示するわけです。

D000 FF-

← なにもデータをメモリに入力していないとき

D000 00-

← FFか00を表示する

D000番地にデータが入力されて記憶されているとすると、そのデータがFFか00の代わりに表示されます。たとえば、D000番地にCDが記憶されていると、次のようになります。

D000 CD-

← データが入力されていると、入力されているデータを表示する

D000番地に入力するデータは、一の右側に入力します。D000番地のメモリに入力するマシン語は3Eですから、3Eと入力します。この場合**RET**キーは叩きません。ただ、3とEのキーを叩くだけです。

mon



*SD000



D000 FF-3E ← D000番地に3Eを入力する

3Eを入力すると、すぐD001番地のFF-を表示します。

mon

*SD000

D000 FF-3E FF- ← D000番地に3Eを入力すると、
すぐD001番地のFF-を表示する

D001番地に入力するマシン語はE9ですから、Eと9のキーを叩きます。

mon

*SD000

D001番地にE9を入力する

D000 FF-3E FF-E9 FF-

このように入力していったら、D000番地からD003番地までに4個のマシン語を入力すると、つぎの実行結果に示すように行が変わって、D000番地のつぎのメモリ番地D004を表示します。そしてD004番地のFFを表示します。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok

mon D000番地に入力

*SD000

D000 FF-3E FF-E9 FF-32 FF-E4 ← D003番地に入力

D004 FF-

↑
D002番地に入力

←
行が変わってD004番地のFFを表示する

Auto go to list home

データを4個入力すると行が変わるのは、横1行40桁表示だからです。横1行80桁にすると、8個のデータを入力したとき行が変わることになります。

横1行40桁の場合 ← 4個のデータを入力すると行が変わる

横1行80桁の場合 ← 8個のデータを入力すると行が変わる

▶ プログラムの入力が終わったら、**RET** キーを叩く

さて、D004番地とD005番地のメモリに、残りのマシン語F6と76を入力すると、マシン語のプログラムの入力は終わりです。そこで**RET** キーを叩きます。**RET** キーを叩く箇所は、D005番地のメモリに76を入力すると、D006番地のFFが表示されますから、そこで**RET** キーを叩くことになります。**RET** キーを叩くと、つぎの実行結果に示すように*印が表示されて、マシン語のコマンド・レベルに戻ります。つまり、つぎのマシン語のコマンドの入力待ちになったわけです。

実行結果

```

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

Ok
mon
*SD000
D000 FF-3E FF-E9 FF-32 FF-E4
D004 FF-F6 FF-76 FF-
*■
    
```

プログラムの入力が終了すると、つぎのメモリ番地のFFを表示するので、そこで **RET** キーを叩く

マシン語のコマンド・レベルに戻る

h
auto
an to
list
run

さて、この場合はマシン語のプログラムを入力し終わったとき、**RET** キーを叩きましたが、**STOP** キーを叩いてもおなじです。つまりマシン語のプログラムの入力が終わったとき叩くキーは、

RET キー

STOP キー

のどちらでもよいということです。

以上が、S コマンドの働きです。

Sコマンドの入力を忘れると

マシン語のプログラムを入力するとき、慣れていないとうっかりしてSコマンドの入力を忘れて、マシン語を入力する最初のメモリ番地だけを入力してしまうことがあります。D000番地の場合は、マシン語のコマンドに、**Dコマンド（リストを取るコマンド）**があるので、つぎのようにSコマンドの入力を忘れて、Dと入力しても変化は起こりません。

mon

*D

Dはリストを取るコマンドなので、
そのままになっている

もちろん、変化が起こらなくても、Dコマンドはリストを取るコマンドなので、マシン語のプログラムの入力できません。

C000番地から入力するというような場合は、Sコマンドの入力を忘れてCと入力すると、つぎの実行結果に示すように、?マークを表示し、その下に*を表示して、Sコマンドが入力されるまで、コマンド・レベルに戻り続けてしまい、マシン語のプログラムの入力ができないことになります。

mon

*C

?

*5

?

*E

?

*S

Sコマンドを忘れて入力するとこのようになる

Sコマンドを入力したところで、最初の
メモリ番地の入力待ちとなる

実行結果

50000000 50000000 50000000 50000000 50000000


?マークは、入力が間違っているときに表示されます。?マークを表示すると、?マークの下に*印が表示されますから、その*印の箇所で、Sコマンドの入力を行うと、最初のメモリ番地の入力待ちとなります。

メモリに記憶させたマシン語のプログラムを画面に表示させる

プログラムを画面に表示させるのはDコマンド

16頁でマシン語のプログラムを入力する方法について説明しました。ここでは入力してメモリに記憶させたものを、画面に表示させる方法について説明します。つまり、ベーシックの場合のLISTの働きと同じです。

まず、MONコマンドを入力してマシン語のコマンド・レベルにします。

mon  マシン語のコマンド・レベルにする
*

Dコマンドは、ダンプ・メモリの意味

マシン語のコマンド・レベルにしたらDコマンドを入力します。Dコマンドは、DUMP MEMORY (ダンプ・メモリ) のことで、メモリの内容を画面に表示させる働きをします。

mon
*D  Dコマンドを入力する

Dコマンドを入力したら、内容を表示させるメモリの最初の番地だけを入力します。ここでは、C000番地を入力してみることにします。

mon
*DC000  C000番地だけを入力して、RETキーを叩く

DコマンドのあとにC000と入力してRETキーを叩くと、つぎの実行結果に示すように、C000番地からC00F番地までのメモリの内容を表示します。

Ok

mon

*DC000  Dコマンドを入力して、C000番地を入力

C000 FF FF FF FF FF FF FF FF C000番地からC00F番地までのメモリの内容を表示する。この場合は、なにも入力していないのでFFを表示する
C008 FF FF FF FF FF FF FF FF
*■

実行結果

このようにDコマンドのあとに、内容を表示させる最初のメモリ番地だけを指定すると、指定したメモリ番地から16個のメモリ番地の内容を表示します。この場合は、C000番地からC00F番地のメモリには、なにも入力していませんから、FF（または00）が表示されます。なにも入力していない場合、メモリに記憶されているデータは、FFか00だからです。

D〈内容を表示させる最初のメモリ番地〉

————16個のメモリの内容を表示する

では、DコマンドのあとにD000番地を入力します。

mon

*DD000  D000番地を入力する

16頁で、D000番地からD005番地までのメモリに入力して記憶させたマシン語のプログラムが、もし電源を切ってしまうなどして消されていなければ、つぎの実行結果に示すように表示されます。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok

mon

*DD000  DコマンドのあとにD000番地を入力する

D000 3E E9 32 E4 F6 76 FF FF

D008 FF FF FF FF FF FF FF FF

*■

↑
入力したマシン語を表示する

↑ auto auto list run

このようにDコマンドのあとに、マシン語のプログラムが記憶されている最初のメモリの番地D000を入力すると、指定したD000番地のメモリから16個目までのメモリの内容を表示しますから、記憶されているマシン語のプログラムとともに、余計なFFも表示されることになります。

16個以上のメモリの内容を表示させるには

Dコマンドのあとにメモリ番地をひとつ指定すると、その番地のメモリから16個目までのメモリの内容を表示しますが、16個のメモリの内容の表示では不足する場合があります。その場合は、

D <内容を表示させる最初のメモリの番地>、<最後のメモリの番地>

というように、Dコマンドのあとに、内容を表示させる最初のメモリの番地と、最後のメモリの番地を指定します。たとえば、つぎのように、

```
mon          内容を表示させる最初のメモリ番地
*D0000, 0050  ← 最後のメモリ番地
```

Dコマンドのあとに0000番地と0050番地を入力してRETキーを叩くと、ROMエリアの最初の0000番地のメモリの内容から、0050番地のメモリの内容までが、つぎの実行結果のように表示されることになります。

mon

*D0000, 0050

0000	F3	31	FF	FF	C3	3B	00	00
0008	C3	6A	00	C3	57	17	AB	F0
0010	C3	59	42	C3	6A	00	DA	0C
0018	C3	A6	40	F3	0B	C3	7E	50
0020	C3	DA	F1	C3	9C	27	88	0C
0028	C3	DD	F1	C3	60	00	46	0C
0030	C3	E0	F1	9F	0F	C3	57	02
0038	C3	E3	F1	AF	32	75	EA	CD
0040	F1	0C	30	C7	18	CD	00	00
0048	00	00	00	00	00	00	00	00
0050	00							

*■

Dコマンドのあとに0000番地と0050番地を指定する

ROMエリアの0000番地から0050番地までのメモリの内容を表示する

実行結果

Auto auto list done

これまで、Dコマンドの働きを見てきてわかるように、ベーシックのLISTコマンドとは、だいぶその働きが違います。ベーシックのLISTコマンドは、「入

力されているプログラムを画面に表示する」という働きですが、マシン語のDコマンドは「メモリの内容を画面に表示する」という働きです。したがって、Dコマンドはマシン語のプログラムが入力されていなくても、指定された番地のメモリの内容を画面に表示するわけです。

表示を上める、表示を再開する、表示を中止する

0000番地から0050番地までのメモリの内容を表示させたように、画面に入るだけのメモリの内容の表示なら問題ありませんが、場合によっては、画面に入り切れないほど、メモリの内容を表示させていくことがあります。

画面に入り切れないほどメモリの内容を表示させると、当然、スクロールされますから、画面に表示されるものは、ほとんど読み取ることができません。そこで、スクロールされている表示を途中で止めます。そのときに使うのが、**ESC**キーです。

ESC ← このキーを叩くとスクロールされている表示が止まる

ESCキーを叩くと、スクロールされている表示が止まります。この場合、**ESC**キーを単に叩くだけです。**RET**キーを叩く必要はありません。

ふたたび、表示を開始するときは、おなじように**ESC**キーを叩きます。

ESC ← 2度目に**ESC**キーを叩くと表示が再開される

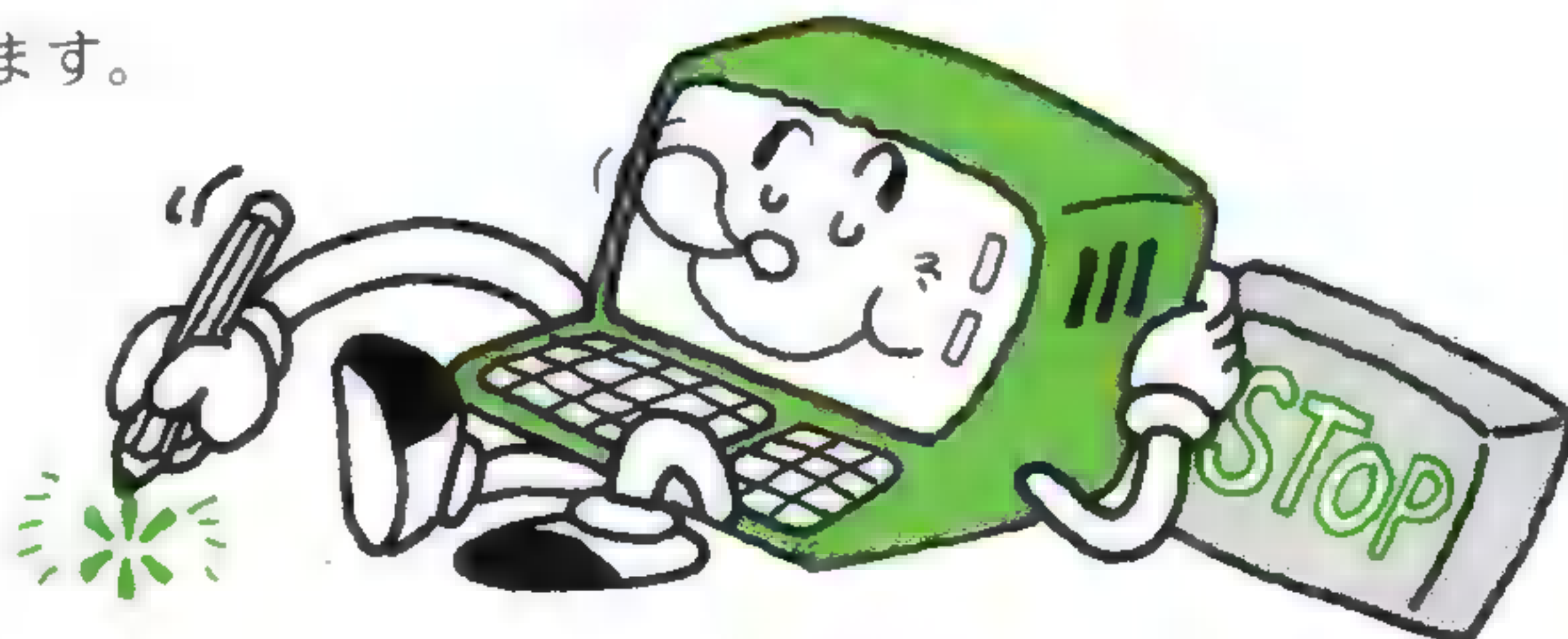
2度目に**ESC**キーを叩くと、表示がふたたび開始されます。

また、表示を途中で中止するときは、**STOP**キーを叩きます。

STOP ← 表示を途中で中止するときは**STOP**キーを叩く

つぎの頁に**ESC**キーを叩いたときの実行結果と、**STOP**キーを叩いたときの実行結果を示します。

ESCキーを叩いたときは、**ESC**キーを叩いた箇所で表示が止まるだけでなくにも表示されませんが、**STOP**キーを叩いたときは、**STOP**キーを叩いた箇所で表示が止まり、その下にマシン語のコマンド・レベルに戻ったという印である*印が表示されます。



ESC キーを押したとき

実行結果

```
mon
*D0000,FFFF
0000 F3 31 FF FF C3 3B 00 00
0008 C3 6A 00 C3 57 17 AB F0
0010 C3 59 42 C3 6A 00 DA 0C
0018 C3 A6 40 F3 0B C3 7E 50
0020 C3 DA F1 C3 9C 27 88 0C
0028 C3 DD F1 C3 60 0D 46 0C
0030 C3 E0 F1 9F 0F C3 57 02
0038 C3 E3 F1 AF 32 75 EA CD
0040 F1 0C 30 C7 18 CD 00 00
0048 00 00 00 00 00 00 00 00
0050 00 00 00 00 ← ESC キーを押したところで、
                        表示を中断する
```

Auto Run Stop List Print

STOP キーを押したとき

実行結果

```
mon
*D0000,FFFF
0000 F3 31 FF FF C3 3B 00 00
0008 C3 6A 00 C3 57 17 AB F0
0010 C3 59 42 C3 6A 00 DA 0C
0018 C3 A6 40 F3 0B C3 7E 50
0020 C3 DA F1 C3 9C 27 88 0C
0028 C3 DD F1 C3 60 0D 46 0C
0030 C3 E0 F1 9F 0F C3 57 02
0038 C3 E3 F1 AF 32 75 ← STOP キーを押したところで、
                        表示を中止する
*■
```

*印を表示して、マシン語のコマンド・レベルに戻る

Auto Run Stop List Print

マシン語のプログラムを実行させる

マシン語のプログラムの実行はGコマンド

16頁でマシン語のプログラムの入力仕方、21頁ではマシン語のプログラムの画面への表示の仕方について説明しました。ここでは、マシン語のプログラムの実行のさせ方について説明します。

この場合は、つぎのプログラムを入力して実行させることにします。

メモリ番地	マシン語	メモリ番地	マシン語	メモリ番地	マシン語
A000	06	A00A	28	A014	A0
A001	12	A00B	77	A015	E1
A002	3E	A00C	23	A016	11
A003	E9	A00D	23	A017	78
A004	21	A00E	10	A018	00
A005	00	A00F	FB	A019	19
A006	F3	A010	C1	A01A	C3
A007	E5	A011	05	A01B	07
A008	C5	A012	CA	A01C	A0
A009	06	A013	1D	A01D	76

左側の4桁の16進数が、マシン語を入力していくメモリ番地です。右側の2桁の16進数が、そのメモリ番地に入力するマシン語です。

マシン語のプログラムの入力は、MONコマンドを入力して、ベーシックのコマンド・レベルからマシン語のコマンド・レベルに切り換えてから、Sコマンドを入力します。そして、最初のメモリ番地をSコマンドのあとに入力します。この場合は、最初の実行するメモリ番地はA000番地ですから、SコマンドのあとにA000を入力することになります。

mon 

*SA000 

S コマンドのあとにA 0 0 0を入力して、**RET** キーを叩くと、A 0 0 0 番地が表示されて、F F - が表示されますから、F F - のあとにA 0 0 0 番地に入力するマシン語0 6を入力します。0 6を入力するとすぐ、つぎのF F - が表示されますから、つぎのマシン語1 2を入力します。このようにして、マシン語のプログラムを入力していきます。


つぎに、マシン語のプログラムを入力した実行結果を示します。

実行結果

*SA000 ← SコマンドのあとにA000番地を入力する

A000	FF-06	FF-12	FF-3E	FF-E9
A004	FF-21	FF-00	FF-F3	FF-E5
A008	FF-C5	FF-06	FF-28	FF-77
A00C	FF-23	FF-23	FF-10	FF-FB
A010	FF-C1	FF-05	FF-CA	FF-1D
A014	FF-A0	FF-E1	FF-11	FF-78
A018	FF-00	FF-19	FF-C3	FF-07
A01C	FF-A0	FF-76	FF-	← 76を入力したつぎのFFで RET キーを叩く

*■ ← マシン語のコマンド・レベルに戻る



マシン語のプログラムの入力、A 0 1 D 番地に7 6を入力すると終わりです。したがって、つぎにF F - が表示されたところで、**RET** キーを叩きます。

FF-76 FF- 

↑ 76を入力 ↑ 76を入力した、つぎのFF-で **RET** キーを叩く

すると、マシン語のコマンド・レベルに戻って、*印が表示されます。

G コマンドはGOのこと

マシン語のプログラムを実行させるには、G コマンドを使います。**G コマンド**は、**GO (ゴー)** ということ、マシン語のプログラムを実行する命令です。G コ

G <プログラムの最初のメモリ番地>

プログラムの入力が
 終わったところで、
 RET キーを叩く

* G * *印のあとに、Gコマンドを入力する

* GA000  ← メモリ 地を入力

A000を入力したら**RET**キーを叩きます。入力ミスなどがなく、プログラムが正しく入力されている場合は、つぎの実行結果に示すように、ハートが画面いっぱいに表示されます。表示のされ方は、ベーシックの場合と違って、ハートが全部いっぺんに表示されます。

実行結果

ふたたび実行させる

さきのマシン語のプログラムを実行させると、ハートを画面いっぱいに表示したまま止まってしまいます。**STOP** キー、**HOME・CLR** キーなど、どんなキーを押してもびくともしません。これはマシン語のプログラムの最後のメモリ番地 **A01D** 番地に、**76** を記憶させているからです。

16進数の **76** は、**HALT**（ホールド）命令です。**HALT** は、英語で停止する、停止させるという意味で、**CPU** の働きを停止させます。

76——HALT（CPUの働きを停止させる）

この**HALT**命令は、ベーシックの**END**文の働きとは多少違いますが、マシン語のプログラムでは**END**文とおなじように、プログラムの終わりや、プログラムの途中で実行を停止するときに使います。

さて、**HALT**命令を使ったときは、**CPU**の働きが停止してしまうので、キーボード上にあるキーを押しても、そのままの状態が続きます。

HALT命令による停止の状態を解くには、リセット・ボタンを押します。すると、つぎに示すように、パソコンに電源を入れた初期状態に戻ります。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok

■ ← ベーシックのコマンド・レベルに戻る

Auto Auto Test Run

メモリに記憶されているマシン語のプログラムは、ベーシックと違ってリセット・ボタンを押しても消えません。したがって、つぎのように、

mon  ← ふたたびマシン語レベルにする

*GA000  ← ふたたび実行する

と入力すると、ふたたびマシン語のプログラムが実行されて、前回とおなじように、ハートが画面いっぱいに表示されます。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok

mon

*GA000

← MON命令でマシン語のコマンド・レベルにして、GA000を入力して[RET]キーを叩くと、ふたたびプログラムを実行する

Auto Run

ベーシックの場合、ENDを省略してもかまいませんが、マシン語の場合、HALT(76) 命令を省略することはできません。エラーになります。

HALT (76) 命令は、実行させたあとCPUが停止してしまい、あとの処置がめんどろですが、ここでは、HALT (76) 命令の働きをおぼえておいてください。



入力したマシン語プログラム を変更する方法

変更するときは、Sコマンドを使う

ベーシックの場合、プログラムを入力したあと、いろいろな方法でプログラムを修正することができます。たとえば、訂正する箇所があるときは、カーソルを訂正する箇所まで移動して訂正するとか、追加するステートメントがあるときは、追加するステートメントに行番号をつけて **RET** キーを叩いておけば、ステートメントが追加できるなどです。

マシン語のプログラムの場合は、ベーシックのような訂正や修正はできません。できるとすれば、メモリに記憶させたマシン語を変更することぐらいで、あとはプログラムの入力のやり直しといってもいいものです。

ここでは、そのメモリに記憶させたマシン語を変更する場合の方法について説明しておくことにします。

26頁で示した、画面いっぱいにハートを表示させるマシン語のプログラムは、それぞれのマシン語に、ひとつひとつメモリ番地を割り当てたように、A000番地から入力して実行させるように作られたものです。

ここでは、そのメモリ番地を無視して、つぎに示すように、D000番地から同じプログラムを入力します。

mon

*SD000 ——— SコマンドのあとにD000番地を入力する

```
D000 FF-06 FF-12 FF-3E FF-E9
D004 FF-21 FF-00 FF-F3 FF-E5
D008 FF-C5 FF-06 FF-28 FF-77
D00C FF-23 FF-23 FF-10 FF-FB
D010 FF-C1 FF-05 FF-CA FF-1D
D014 FF-A0 FF-E1 FF-11 FF-78
D018 FF-00 FF-19 FF-C3 FF-07
D01C FF-A0 FF-76 FF- ———
```

*■

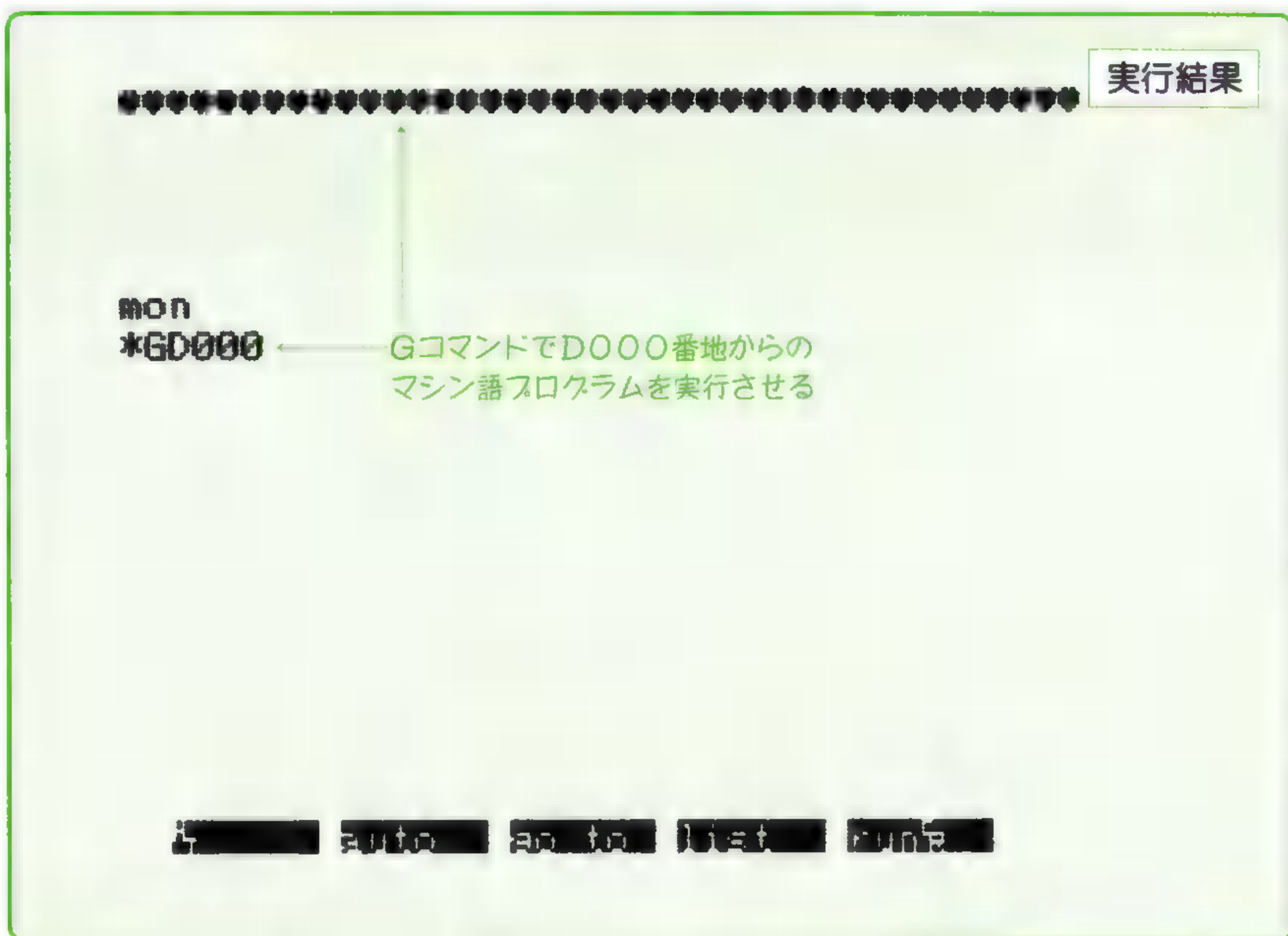
RET キーを叩く

実行結果

プログラムをD000番地
から記憶させる

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

このように、A 0 0 0 番地から入力して実行するように作られているプログラムを、D 0 0 0 番地から入力して実行させるとすると、つぎの実行結果に示すように、横に一行ハートを表示すると、ストップしてしまいます。



実行がストップしたのは、D 0 1 2 番地から D 0 1 4 番地に入力した、

C A 1 D A 0

と D 0 1 A 番地から D 0 1 C 番地に入力した、

C 3 0 7 A 0

に問題があるからです。C A と C 3 は、JP 命令といって、「ジャンプせよ」という命令です (116 頁参照)。1 D A 0 と 0 7 A 0 は、JP 命令でジャンプしていくさきのメモリ番地です。

JP 命令で、A 0 1 D 番地、または A 0 0 7 番地にジャンプしても、そのメモリ番地にはなにも入力していません。なぜなら、この場合は、D 0 0 0 番地から入力して実行させているからです。そのため、実行がそこでストップしてしまうわけです。

そこで、プログラムが正しく実行するように、JP 命令によるジャンプさきを A 0 1 D 番地から D 0 1 D 番地に、また、A 0 0 7 番地を D 0 0 7 番地に変更しなければなりません。

入力したマシン語プログラムを変更する方法

変更するには、Dコマンドを使ってメモリに入力したマシン語を画面に表示させなければなりませんが、このままではDコマンドを使うことができません。

そこで、リセット・ボタンを押します。リセット・ボタンを押すと、パソコンに電源を入れた初期状態に戻ります。つまり、ベーシックのコマンド・レベルです。

ベーシックのコマンド・レベルに戻ったら、MON命令を入力してRETキーを叩き、マシン語のコマンド・レベルにします。

mon 

*

そして、Dコマンドと最初のメモリ番地と最後のメモリ番地を入力します。

mon

*DD000, D01F 

これで、D000番地からD01F番地に記憶されているマシン語のプログラムが、つぎの実行結果に示すように表示され、その下に*印が表示されます。

mon

*DD000, D01F ———— DコマンドとD000とD01F番地を

D000 06 12 3E E9 21 00 F3 E5 入力して、メモリに入力した
D008 C5 06 28 77 23 23 10 FB マシン語を画面に表示する

D010 C1 05 CA 1D A0 E1 11 78

D018 00 19 C3 07 A0 76 FF FF

*■

実行結果

Auto Run Reset Run

この*印の横に16頁で説明したSコマンドと、変更する必要があるマシン語が記憶されているメモリ番地を入力します。この場合は、変更する箇所はD014番地のA0と、D01C番地のA0です。このふたつのA0をD0に変えればよいわけです。

まず、Sコマンドを入力して、最初のA0が記憶されているメモリ番地D014を入力します。

*SD014 

訂正するマシン語が記憶されている
メモリ番地を入力する

D014を入力して、**RET**キーを叩くと、D014番地を表示して、そのメモリに記憶されているマシン語、つまり、A0を表示します。

*SD014 

D014 A0-

D014番地のA0を表示する

そこで、-の横にD0を入力します。

D014 A0-D0 E1-

A0-の横にD0を入力する

変更する必要のないところはスペースキーで飛ばす

D0を入力すると、つぎのメモリ番地D015に記憶されているマシン語E1を表示します。

このE1は、変更する必要がありませんから、キーボードの一番下にあるスペースキーを叩きます。すると、つぎのメモリ番地D016番地に移って、D016番地に記憶されているマシン語11を表示します。

D014 A0-D0 E1- 11-

訂正の必要のない箇所は
スペースキーを叩いて飛ばす

D016番地のマシン語11も変更する必要がありませんから、スペースキーを叩きます。


このように変更する必要のないメモリ番地は、スペースキーを叩いて飛ばしていきます。スペースキーを叩いて、飛ばしたメモリ番地に記憶されているマシン語は、そのまま記憶されています。

さて、もうひとつのA0が記憶されているD01C番地まできたら、A0をD0に変更します。

D01C A0-D0

A0-の横にD0を入力する

D01C番地のA0をD0に変更すると、この場合の変更は終わりですから、つぎのD01D番地の76が表示されたところで、**RET**キーを叩きます。

D01C A0-D0 76 

つぎのメモリ番地の内容を表示
したところで **RET** キーを叩く

つぎの頁に、この変更の実行結果を示します。



実行結果

```

mon
*DD000,D01F
D000 06 12 3E E9 21 00 F3 E5
D008 C5 06 28 77 23 23 10 FB
D010 C1 05 CA 1D A0 E1 11 78
D018 00 19 C3 07 A0 76 FF FF
*SD014 Sコマンドを入力して、変更するメモリ番地を入力
D014 A0-D0 E1- 11- 78-
D018 00- 19- C3- 07-
D01C A0-D0 76-
*

```

AOをDOに変更する
変更しないところは、スペースキーで飛ばす

auto
auto
list
run

この場合、D01D番地の76のところでRETキーを叩き忘れて、スペースキーを叩き続けると、つぎのように、移動していくので注意してください。

実行結果

```

mon
*DD000,D01F
D000 06 12 3E E9 21 00 F3 E5
D008 C5 06 28 77 23 23 10 FB
D010 C1 05 CA 1D A0 E1 11 78
D018 00 19 C3 07 A0 76 FF FF
*SD014
D014 A0-D0 E1- 11- 78-
D018 00- 19- C3- 07-
D01C A0-D0 76- FF- FF-
D020 FF- FF- FF- FF-
D024 FF- FF- FF- FF-
*

```

スペースキーを叩き続けると、移動しつづける

auto
auto
list
run

まえのメモリ番地に戻すときは

スペースキーを叩いた場合は、メモリ番地がさきに移動していきましたが、メモリ番地をまえに戻したいという場合もあります。そのようなときは、

DEL キー ← まえのメモリ番地に戻すとき使う

を叩きます。

つぎに示す実行結果は、D012番地でD0と入力すべきところを、A0と入力したので、D013番地のFFーで、**DEL** キーを押した場合です。

mon

*SD000

D000 FF-06 FF-12 FF-21 FF-00

D004 FF-F3 FF-E5 FF-C5 FF-06

D008 FF-28 FF-77 FF-23 FF-23

D00C FF-10 FF-FB FF-C1 FF-05

D010 FF-CA FF-1D FF-A0 FF- ← **DEL** キーを押す

D012 A0- ← ひとつまえのメモリ番地に戻って、
そのメモリに記憶されているマシン
語を表示する

*5x

実行結果

auto auto to list run

うえの実行結果を見るとわかると思いますが、

D010 FF-CA FF-1D FF-A0 FF- ← **DEL** キーを押す

D012 A0- ← D012番地に戻って、A0を表示する

D013番地のFFが表示されたところで、**DEL** キーを押したので、ひとつまえのD012番地に戻って、新たにD012番地を表示して、D012番地のマシン語A0を表示したわけです。

このD012番地のA0ーに対して、D0を入力すると、**DEL** キーを押したD013番地に移動して、D013番地のFFーを表示します。

D012 A0-D0 FF- ← **DEL** キーを押したD013番地に移る

つぎにその、実行結果を示します。

mon
*SD000
D000 FF-06 FF-12 FF-21 FF-00
D004 FF-F3 FF-E5 FF-C5 FF-06
D008 FF-28 FF-77 FF-23 FF-23
D00C FF-10 FF-FB FF-C1 FF-05
D010 FF-CA FF-1D FF-A0 FF-
D012 A0-D0 FF-

実行結果

DEL キーを押す

D012番地のA0に対してD0を入力すると、D013番地に移って、FFを表示する

auto go to list run

この **DEL** キーを押し続けると、オートリピートが働いて、つぎの実行結果に示すように、つぎつぎにまえのメモリ番地に移動していきます。

mon
*SD000
D000 FF-06 FF-12 FF-3E FF-E9
D004 FF-21 FF-00 FF-F3 FF-E5
D008 FF-C5 FF-06 FF-28 FF-77
D00C FF-23 FF-23 FF-10 FF-FB
D010 FF-C1 FF-05 FF-CA FF-1D
D014 FF-A0 FF-E1 FF-
D015 E1-
D014 A0-
D013 1D-
D012 CA-
D011 05-
D010 C1-

実行結果

DEL キーを押す

まえのメモリ番地に戻る

DEL キーを押し続けると
どんどんまえのメモリ番地に戻っていく

auto go to list run

間違ってマシン語(16進数)に入力したとき

マシン語は16進数で書き表わされています(16進数については91頁参照)。そこで、キーを押し間違えて、16進数に入力すると、たとえばGとかH、SとかXを入力すると、つぎの実行結果に示すように?マークを表示し、*印がその下に表示されてマシン語のコマンド・レベルに戻ってしまいます。

mon

*SD000

D000 FF-06 FF-12 FF-21 FF-00

D004 FF-F3 FF-E5 FF-C5 FF-06

D008 FF-28 FF-77 FF-23 FF-23

D00C FF-10 FF-FB FF-S ← 16進数にないSを入力

? ← ?マークを表示してマシン語の

*■ コマンド・レベルに戻る

実行結果

← auto go to list run

このようなときは、つぎのように、*印の横にSコマンドを入力して、入力を間違えたメモリ番地を指定してもかまいません。

?

*SD00E  ← Sコマンドのあとに、入力を間違えたメモリ番地を入力する

すると、つぎのように、間違えて入力したメモリ番地を表示して、FF-(あるいは00-)を表示しますから、

D00E FF- ← 入力を間違えたメモリ番地のFFを表示する

正しいマシン語を入力します。

ところで、メモリ番地はとびとびに表示されています。したがって、メモリの番地の表示の仕方に慣れていないと、何番地のメモリ番地を指定してよいのか、さっぱりわからないということもあります。

入力したマシン語プログラムを変更する方法

そこで、このように入力を間違えたメモリ番地を指定しなくてもよい方法があります。つまり、*印の横に単にSと入力して、**RET** キーを押せばよいのです。

*S  Sコマンドを入力して **RET** キーを叩く

これで、つぎの実行結果に示すように、入力を間違えたメモリ番地に戻ります。

```
mon
*SD000
D000 FF-06 FF-12 FF-21 FF-00
D004 FF-F3 FF-E5 FF-C5 FF-06
D008 FF-28 FF-77 FF-23 FF-23
D00C FF-10 FF-FB FF-S
?
*S Sを入力して RET キーを叩くと
D00E FF- 入力ミスをおこしたメモリ番地に戻る
```

実行結果

auto go to list run



コマンドのあとのメモリ番地の 入力を間違えたときは

メモリ番地の入力を間違えたときは「STOP」キー

マシン語を入力するときに使うSコマンド、メモリに記憶させたマシン語を画面に表示させるDコマンド、あるいはマシン語のプログラムを実行させるGコマンドを使う場合、次に示すように、そのあとにメモリ番地を入力しますが、

*SD000

*DD000, D01C

*GD000

S、D、Gコマンドのあとには
メモリ番地を入力する

このメモリ番地を間違えて入力することがあります。

SコマンドやDコマンドの場合、メモリ番地を間違えて入力して、**RET**キーを叩いても問題はありませんが、Gコマンドの場合は、暴走して、プログラムを破壊してしまいます。

そこでここでは、コマンドのあとに入力するメモリ番地の入力を間違えたときどうすればよいか、その方法について説明します。

方法といっても簡単なことで、メモリ番地の入力を間違えたことに気がついたところで、**STOP**キーを押せばよいだけです。

たとえば、マシン語のプログラムがD000番地から入力されていたとします。このマシン語のプログラムを実行させる場合、まずGコマンドを入力して、そのあとにマシン語のプログラムが記憶されているメモリ番地D000を入力します。

mon

*GD000 ← 正しい入力

そして**RET**キーを叩けば、D000番地から記憶されているマシン語のプログラムが実行されますが、GコマンドのあとにD000と入力するところを、うっかりしてGコマンドのあとにD003と入力したとします。

mon

*GD003 ← 間違えて入力

コマンドのあとのメモリ番地の入力を間違えたときは

このまま、**RET** キーを叩くと暴走します。暴走したらプログラムは破壊されますが、パソコンが壊れるわけではないので、パソコンのリセットキーを押すと、ベーシックのコマンド・レベルに戻るので、ふたたび**MON** コマンドを入力して、マシン語のコマンド・レベルにすれば、なにも心配することはありませんが、メモリ番地の入力を間違えたことに気がいたら、メモリ番地の入力をしなおしたほうが入力したプログラムの保護にもなります。

G コマンドのあとに **D003** を入力すると、カーソルは **D003** のあとで点滅していますから、その位置で **STOP** キーを押します。

***GD003** ———— この位置でカーソルが点滅

STOP キーを押すと、つぎに示すように、つぎの行に ***** 印を表示します。

GD003 ———— **CTRL** キーと **B** キーを一緒に押す

***** ———— ふたたび ***** 印を表示する

そこで、ふたたび表示された ***** に対して、G コマンドと、正しいメモリ番地を入力します。これが、コマンドのあとに入力するメモリ番地の入力を間違えたときの訂正の方法です。つぎに、その実行結果を示します。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok
mon
***GD003** ———— **STOP** キーを押す
GD000** ———— ふたたび表示した ** 印に対して
G コマンドと正しいメモリ番地
を入力する

auto go to list done

S コマンドと D コマンドの場合も、メモリ番地の入力をやりなおすことは G コマンドの場合とおなじです。

S コマンドのあとに入力したメモリ番地が間違っていることを、**RET** キーを叩くまえに気がついたときは、さきに説明した G コマンドのときのように、**STOP** キーを押して新しく * 印を表示させて、S コマンドと正しいメモリ番地を入力しますが、もし **RET** キーを叩いてしまったあと、メモリ番地の入力間違いに気がついたときは、最初に表示された FF- の箇所で、**STOP** キーを押します。

mon

*SC000

C000 FF- **STOP** キーを押す

すると、つぎの実行結果に示すように、ふたたび * 印を表示しますから、その * 印に対して、S コマンドと正しいメモリ番地を入力して、**RET** キーを叩きます。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok

mon

*SC000 違ったメモリ番地を入力したので、**RET** キーを叩く

C000 FF- この位置で **STOP** キーを押す

*SD000 新しく表示された * 印に対して、S コマンドと

D000 FF- 正しいメモリ番地を入力する

auto auto list run

D コマンドのあとに入力したメモリ番地が間違っていることを、**RET** キーを叩くまえに気がついたときは、G コマンドのときのように **STOP** キーを押して、新しく * 印を表示させて、D コマンドと正しいメモリ番地を入力します。

コマンドのあとのメモリ番地の入力を間違えたときは

また、もし間違えたメモリ番地のまま **RET** キーを叩いたときは、そのメモリ番地に記憶されているマシン語を表示して、そのあとに * 印を表示しますから、表示した * 印に対して、D コマンドと正しいメモリ番地を入力すればよいわけです。

間違って入力したメモリ番地に、なにも入力していないときは、つぎの実行結果に示すように、FF か 00 を表示します。そして、FF か 00 を表示したあとに * 印を表示しますから、ふたたび D コマンドと正しいメモリ番地を入力して **RET** キーを叩きます。

mon

*DC000,C01C

C000 FF FF FF FF FF FF FF FF

C008 FF FF FF FF FF FF FF FF

C010 FF FF FF FF FF FF FF FF

C018 FF FF FF FF FF

*DD000,D01C

D000 06 12 3E E9 21 00 F3 C5

D008 06 28 77 23 23 10 FB C1

D010 05 CA 1D D0 E1 11 78 00

D018 19 C3 07 D0 76

*■

実行結果

← なにも記憶していないと
FF か 00 を表示する

*を表示したらその ■ に対して、
D コマンドと正しいメモリ ■ 地を
入力する

to auto go to last run



マシン語のプログラムを カセットテープに記録する

プログラムをカセットテープに記録するにはWコマンド

ベーシックで書いたプログラムが完成すると、カセットテープに記録して保管します。おなじようにマシン語のプログラムも、カセットテープに記録して保管することができます。

ベーシックのプログラムをカセットテープに記録する場合は、**CSAVE**を使います。これに対してマシン語のプログラムをカセットテープに記録する場合は、**Wコマンド**を使います。Wコマンドは、**WRITE TAPE**（ライト・テープ）のことで、メモリに記憶されているマシン語をカセットテープに出力する働きをします。

Wコマンドのあとにはメモリ番地を置く

また、ベーシックのプログラムをカセットテープに記録する場合は、CSAVEのあとに“ファイル・ネーム”を置きますが、マシン語のプログラムをカセットテープに記録する場合は、つぎに示すようにWコマンドのあとにマシン語のプログラムを記憶しているメモリ番地を置きます。

W〈プログラムを記憶している最初のメモリ番地, 最後のメモリ番地〉

では、どのようにしてカセットテープに記録するのか実際にマシン語のプログラムをカセットテープに記録することにします。

マシン語のプログラムをカセットテープに記録する場合はベーシックのコマンド・レベルにある状態の場合とマシン語のコマンド・レベルにある状態の場合の2通りが考えられますが、いずれの場合も、*印のあとにWコマンドと、プログラムが記憶されているメモリ番地を入力するというものですので、ここでは、ベーシックのコマンド・レベルにある状態の場合を取り上げて説明することにします。


マシン語のプログラムをカセットテープに記録する

ここで使うマシン語のプログラムは31頁で取り上げた画面いっぱいにハートを表示するプログラムです。このプログラムは、つぎに示すように、D000番地からD01D番地までに記憶されています。

```
*DD000,D01D
D000 06 12 3E E9 21 00 F3 E5
D008 C5 06 28 77 23 23 10 FB
D010 C1 05 CA 1D D0 E1 11 78
D018 00 19 C3 07 D0 76
*■
```

このマシン語のプログラムは最後にHALT（76H）命令が使われているので、プログラムが間違いなく実行するかどうかを確かめると、リセット・ボタンを押して、いったんベーシックのコマンド・レベルに戻さなければなりません。そのため、Wコマンドを入力するには、MON命令を入力してマシン語のコマンド・レベルにします。

mon

*  マシン語のコマンド・レベルにする

そのうえで、つぎに示すように、Wコマンドを入力し、つづいて、マシン語のプログラムが記憶されている最初のメモリ番地D000番地と、最後のメモリ番地D01D番地を入力します。

mon

* WD000, D01D



Wコマンドのあとに最初のメモリ番地と最後のメモリ番地を置く

このように入力したら、RETキーを叩くまえに、ベーシックのプログラムをカセットテープに記録する場合とおなじように、データレコーダにカセットテープをセットして、RECボタンとPLAYボタンを押します。

それから、RETキーを叩きます。RETキーを叩くと、カーソルはWコマンドのまえにある*印のうえに移動して、*印のうえで点滅します。

mon  この位置にカーソルが移動して点滅する

* WD000, D01D

カセットテープへの記録が終了すると、つぎの行に*印が表示されてマシン語のコマンド・レベルに戻ります。

不幸にして、カセットテープへの記録が失敗したときは？マークが表示され、つぎの行に*印が表示されてマシン語のコマンド・レベルに戻ります。

カセットテープへの記録が失敗したときは、もう一度記録のやり直しです。
つぎに、カセットテープへの記録が正常に行われた場合と、失敗に終わった場合の実行結果を示します。

▶ 記録が正常に行われたとき

実行結果

```
NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

Ok
mon
*WD000,D01D
*■ ← 記録が正常に行われたときは
      *印が表示されてマシン語の
      コマンド・レベルに戻る
```

auto auto list print

▶ 記録が失敗したとき

実行結果

```
NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

Ok
mon
*WD000,D01D
? ← 記録が失敗したときは?マークが表示される
* ← そして*印が表示されて、マシン語のコマンド・レベルに戻る
```

auto auto list print

ベリファイを行うには

ベリファイを行うには、LVコマンド

ベーシックのプログラムを、パソコンのメモリからカセットテープに記録したあと、プログラムが間違いなくカセットテープに記録されたかどうか、ベリファイを行います。マシン語のプログラムをカセットテープに記録したあともベリファイを行います。

ベーシックのプログラムの場合のベリファイは、CLOAD? を使って行いますが、マシン語のプログラムの場合は、LVコマンドを使って行います。

LVコマンドは、VERIFY TAPE (ベリファイ・テープ) のことで、テープに記録したマシン語のプログラムと、メモリに記憶されているマシン語のプログラムが同じものかどうかを比較する働きをします。

ベリファイを行うには、LV

ベーシックのプログラムの場合、ベリファイを行うには、CLOAD? のあとに“ファイル・ネーム”を入力しますが、マシン語のプログラムの場合はLVコマンドを入力して、RET キーを叩くだけです。

LV 

では、どのようにしてベリファイを行うのか、実際にベリファイを行ってみることにします。

さて45頁で、Wコマンドを使って、マシン語のプログラムをカセットテープに記録しました。マシン語のプログラムのカセットテープへの記録が正常に行われると、*印が表示されてマシン語のコマンド・レベルに戻りました。

mon


*WD000, D01D

* ——— マシン語のプログラムの記録が正常に行われると*印を表示する

この*印の横に、LVコマンドを入力します。

mon

*WD000, D10D

*LV  ← *印の横にLVを入力する

LVを入力したら **RET** キーを叩きますが、そのまえにマシン語のプログラムを記録した先頭より少し前の位置までカセットテープを巻き戻し、データレコーダの **PLAY** ボタンを押します。それから、**RET** キーを叩きます。

RET キーを叩くと、カーソルはLVの左の*印の上に移動して点滅します。

mon

*WD000, D01D

 LV **RET** キーを叩くと、カーソルは*印の上に移動して点滅する

ベリファイを行って、カセットテープに記録したマシン語のプログラムとメモリのなかに記憶されているマシン語のプログラムが一致すると、つぎの実行結果に示すように、つぎの行に*印を表示してマシン語のコマンド・レベルに戻ります。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft


実行結果

Ok

mon

*WD000, D01D

*LV

*  ← カセットテープのマシン語のプログラムと、
メモリのなかのマシン語のプログラムが一致
すると*印を表示する

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 25

ベリファイを行うには

カセットテープに記録したマシン語のプログラムとメモリのなかに記録されているマシン語のプログラムが一致しなかった場合も、カセットテープに記録したマシン語のプログラムに異常があるわけですから、もう一度カセットテープへの記録のやり直しとなります。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok
mon
*WD000,D01D
*LU
?
*

カセットテープのマシン語のプログラムとメモリのなかのマシン語のプログラムが一致しないと、?マークを表示する

Go to Go to Next Page



カセットテープに記録したプログラムをメモリのなかに読み込むには

メモリのなかに読み込むには、Lコマンド

カセットテープに記録したマシン語のプログラムは必要に応じてパソコンのメモリに読み込んで使います。

ベーシックのプログラムの場合は、カセットテープからパソコンのメモリのなかにプログラムを読み込むときはCLOADを使って行いますが、マシン語のプログラムの場合は、Lコマンドを使って行います。

Lコマンドは、LOAD TAPE（ロード・テープ）のことで、カセットテープに記録されているマシン語のプログラムをパソコンのメモリのなかに読み込む働きをします。

メモリのなかに読み込むには、L

ベーシックのプログラムをメモリに読み込む場合、CLOADのあとに“ファイル・ネーム”を置きましたが、マシン語のプログラムをメモリに読み込むには、Lコマンドを入力して、**RET**キーを叩くだけです。

L 

ではどのようにしてマシン語のプログラムをメモリに読み込むのか、実際にカセットテープに記録されているマシン語のプログラムをメモリのなかに読み込んでみることにします。

45頁でWコマンドを使ってマシン語のプログラムをカセットテープに記録しましたから、ここではそのマシン語のプログラムをメモリに読み込むことにします。まず、パソコンの電源を切って、メモリに記憶されているプログラムを消去します。そして、MONコマンドを入力してマシン語のコマンド・レベルにします。

mon

* ← —— マシン語のコマンド・レベルにする

カセットテープに記録したプログラムをメモリのなかに読み込むには

*印が表示されたら、*印の横にLコマンドを入力します。

mon

* L  *印の横にLを入力する

Lコマンドを入力したら、**RET** キーを叩くまえに、カセットテープを巻き戻すなり、先送りするなどしてメモリに読み込むマシン語のプログラムの先頭より少し前の位置に合わせます。これは、ベーシックのプログラムをメモリに読み込む場合とおなじ要領です。

そして、データレコーダの**PLAY** ボタンを押すとともに、**RET** キーを叩きます。すると、カーソルは、Lコマンドの左の*印のうえに移動して点滅します。

mon

 L  カーソルは*印のうえで点滅する

カセットテープのマシン語のプログラムがメモリのなかに読み込まれると、つぎの行に*印が表示されて、マシン語のコマンド・レベルに戻ります。

つぎに実行結果を示します。


NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft


実行結果

Ok

mon

*L  Lコマンドを入力して**RET** キーを叩く

*  メモリへのプログラムの読み込みが
終わると*印を表示してマシン語の
コマンド・レベルに戻る

 auto auto list run

読み込まれたプログラムが記憶されるメモリ番地は？

さて、カセットテープのマシン語のプログラムはカセットテープからメモリのなかに読み込まれて、どのメモリ番地から記憶されたのでしょうか。

45頁でWコマンドを使って、メモリに記憶されているマシン語のプログラムをカセットテープに記録するとき、マシン語のプログラムはD000番地からD01D番地に記憶されていました。そこで、つぎのようにして、カセットテープにマシン語のプログラムを記録させました。

*WD000, D01D

このメモリ番地はマシン語のプログラムと一緒にカセットテープに記録されます。したがってLコマンドでメモリに読み込まれるマシン語のプログラムは、Wコマンドでカセットテープに記録したときに指定した番地に読み込まれることになります。

では確かに、いま説明したとおりであるかどうか、Dコマンドを使ってメモリ番地を指定して、メモリに読み込まれたマシン語のプログラムを画面に表示させてみることにします。つぎに示すのが、その実行結果です。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok
mon
*L

*DD000, D01D ← Wコマンドで指定した番地を、Dコマンドで指定する

D000 06 12 3E E9 21 00 F3 E5
D008 C5 06 28 77 23 23 10 FB
D010 C1 05 CA 1D D0 E1 11 78
D018 00 19 C3 07 D0 76

LコマンドでD000番地から
D01D番地に読み込まれたマ
シン語のプログラム

*■

h auto auto list run

なおカセットテープからのプログラムの読み込みが失敗したときは、?マークが表示され、*印が表示されてマシン語のコマンド・レベルに戻ります。

ベーシックとマシン語でできたプログラムをカセットテープに記録する

ベーシックを記録するにはCSAVE、マシン語はW

いろいろな雑誌などにベーシックとマシン語で作られたプログラムが掲載されています。ベーシックとマシン語で作られたプログラムをキーボードのキーを叩いて苦労してパソコンに入力しても、そのプログラムをカセットテープに保管する方法を知らないと、ふたたびそのプログラムを使うとき、また同じ苦労をして、パソコンにプログラムを入力しなければなりません。そこで、ここでベーシックとマシン語でできたプログラムをカセットへ記録する方法について説明しておくことにします。

方法といっても、非常に簡単なことです。ベーシックのプログラムをカセットテープに記録するには、

CSAVE “ファイルネーム”

を使って行います。

マシン語のプログラムをカセットテープに記録するには、45頁で説明したように、

W 〈最初のメモリ番地、最後のメモリ番地〉

を使って行います。

ベーシックとマシン語で作られたプログラムをカセットテープに記録するには、このふたつのコマンドを使って行えばよいのです。

では、実際にベーシックとマシン語でできたプログラムを、カセットテープに記録させてみることにします。

ここでは、つぎの頁のうえに示すプログラムを使うことにします。このプログラムは、最初がベーシックでできています。つまり、行番号10から150までです。行番号150のEND文のあとにつづいているのがマシン語のプログラムです。つまり、メモリ番地D000からD04Bまでの16進数の部分です。

▶ ベーシックとマシン語でできたプログラム

```

10 DEF USR=&HD000
20 WIDTH 80,25:CONSOLE,,0
30 DIM MM(5)
40 PRINT CHR$(12)
50 PRINT "  ▲ ▲  "
60 PRINT "  ■■■■■  "
70 PRINT "  ●▼●●  "
80 GET@ (0,1)-(6,3),MM
90 PRINT CHR$(12)
100 FOR I=0 TO 40
110 PUT@ (I,22)-(I+6,24),MM
120 NEXT
130 A=USR(0)
140 LOCATE 0,15
150 END

```

ベーシックの
プログラム

```

D000 21 7B FD 06 14 3E EC 77
D008 16 FF 3E 20 D3 40 0E FF
D010 0C C2 10 D0 15 C2 0A D0
D018 3E 00 D3 40 3E 00 77 11
D020 78 00 ED 52 10 DF 06 0C
D028 11 40 D0 21 9A F3 1A 77
D030 23 13 10 FA C9 FA 76 FF
D038 FF FF FF FF FF FF FF FF
D040 B7 B6 B2 BA DE CC DF DB
D048 B8 DE D7 D1 00 00 00 00

```

マシン語の
プログラム

このマシン語のプログラムはサブルーチンになっていて、メインルーチンのベーシックのプログラムから、このマシン語のプログラムを呼び出しているのです。

ベーシックとマシン語でできたプログラムをカセットテープに記録する

ベーシックとマシン語でできたプログラムは、このプログラムに示すように、ベーシックでできた部分とマシン語でできた部分とが、ふたつに別かれているのがふつうです。

さて、このプログラムのカセットテープへの記録です。このプログラムは、すでにパソコンに入力がすすんでいるものとします。

まず、ベーシックのプログラムを記録する

このようにベーシックとマシン語でできているプログラムをカセットテープに記録するには、まずベーシックでできているプログラムをカセットテープに記録します。ベーシックのプログラムをカセットテープに記録するには、さきに表示したように、ファイル・ネームが必要です。ここではファイル・ネームを簡単にして“a a”とします。したがって、

CSAVE “a a” ベーシックのプログラムをセーブする

と入力して、データレコーダにカセットテープをセットして、**REC** ボタンと **PLAY** ボタンを押します。それから **RET** キーを叩きます。ベーシックのプログラムが正常に記録されるとOKを表示しますから、OKを表示したら、データレコーダの **STOP** キーを押して、データレコーダを止めます。

つぎに、マシン語のプログラムを記録する

OKが表示されたら、MONコマンドを入力してマシン語のコマンド・レベルにします。

mon

* · —— マシン語のコマンド・レベルにする

マシン語のコマンド・レベルにしたらWコマンドを入力します。そして最初のメモリ番地と最後のメモリ番地を入力します。この場合マシン語のプログラムは、D 0 0 0 番地からD 0 4 B 番地に記憶されているので、最初のメモリ番地D 0 0 0 番地と、D 0 4 B 番地を入力します。

mon

*WD 0 0 0, D 0 4 B · Wコマンドのあとに最初のメモリ番地と最後のメモリ番地を入力する

RET キーを叩くまえに、少しカセットテープを先送りして、ベーシックのプログラムとの間を少し離します。そのうえで、データレコーダの **REC** ボタンと **PLAY** ボタンを押します。そして、**RET** キーを叩きます。

すると、カーソルはWコマンドの横の*印のうえに移動して点滅します。

mon

カーソルがこの位置で点滅する

*WD000, D04B

マシン語のプログラムが正常に記録されると、次の行に*印が表示されます。

つぎにベーシックのプログラムのカセットテープへの記録と、マシン語のプログラムのカセットテープへの記録の状態の実行結果を示します。

```
csave"aa"  ベーシックのプログラムの記録
Ok
mon  マシン語のコマンド・レベルにする
*WD000, D04B  マシン語のプログラムの記録
*5<  CTRLキーとBのキーを一緒に押して
Ok  ベーシックのコマンド・レベルに戻す
cload?"aa"  ベーシックのプログラムのベリファイ
Found:aa
Ok
mon  マシン語のプログラムのベリファイ
*LU
*5<  CTRLキーとBのキーを一緒に押して
Ok  ベーシックのコマンド・レベルに戻す
■
```

実行結果

Auto Auto List Run

ベリファイを行うには

うえの実行結果に示したように、ベリファイも行っています。

この場合は、最初にベーシックのプログラムのベリファイを行い、つぎにマシン語のベリファイを行っています。

そうするにはマシン語のプログラムが正常に記録されたという*印が表示されたら、*印の箇所で、CTRLキーとBのキーを一緒に押してマシン語のコマンド・レベルからベーシックのコマンド・レベルに戻します。そして、カセットテープをベーシックのプログラムの位置の少し先まで巻き戻して、

CLOAD? "a a" ベーシックのプログラムのベリファイを行う

を入力します。それから、データレコーダのPLAYボタンを押して、RET

ベーシックとマシン語でできたプログラムをカセットテープに記録する

キーを叩き、ベーシックのプログラムのベリファイを行います。

ベーシックのプログラムのベリファイが終了してOKが表示されたら、データレコーダの **STOP** キーを押して、データレコーダを止めます。

つぎに、MONコマンドを入力してマシン語のコマンド・レベルにします。

mon

* ← マシン語のコマンド・レベルにする

マシン語のコマンド・レベルにしたら、*の横にLVコマンドを入力します。

mon

* LV ——— マシン語のプログラムのベリファイを行う

LVコマンドを入力したら、マシン語のプログラムはベーシックのプログラムのあとに記録されているので、そのままデータレコーダの **PLAY** ボタンを押して、**RET** キーを叩きます。

すると、カーソルはLVコマンドの横の*印のうえに移動して点滅します。

mon

✱ LV ← カーソルが*印のうえで点滅する

マシン語のプログラムのベリファイが終了すると、つぎの行に*印を表示します。

ベーシックとマシン語でできたプログラムはベーシックのコマンド・レベルで実行させますから、*印が表示されたら **CTRL** キーと **B** のキーを一緒に押して、ベーシックのコマンド・レベルに戻します。

プログラムの説明

では、このプログラムについて簡単に説明しておきます。

このプログラムを実行させると、つぎの実行結果に示すように、まず、車が画面の下、左端に表示されます。

実行結果



← 画面の左端に車が表示される


```

30 DIM MM(5)
40 PRINT CHR$(12)
50 PRINT "  ▲ ▲  "
60 PRINT "  ■■■■  "
70 PRINT "  ●▼●  "
80 GET@ (0,1)-(6,3),MM

```

実行結果のように車を画面の左端に表示するのは、まず、行番号50から70までのPRINT文で画面に車を表示します。このPRINT文で表示された車は行番号80のGET@文で、行番号30のDIM文で配列の宣言をして確保された配列MM(5)に記憶されます。配列に記憶された車は、行番号110のPUT@文で取り出されて表示されます。

```

90 PRINT CHR$(12)
100 FOR I=0 TO 40
110 PUT@ (I,22)-(I+6,24),MM
120 NEXT

```

行番号110のPUT@文は、行番号100と120のFOR～NEXT文で囲まれています。この行番号100のFOR文の変数Iの値がPUT@文のカッコのなかのIに送られてきて、PUT@文で配列MMに記憶した車を取り出して、表示する位置の指定となります。

行番号110のFOR文の変数Iの初期値は0です。したがって、この0がPUT@文のカッコのなかのIに送られてきますから、

PUT@ (0, 22) - (0+6, 24), MM

となって、画面の左端に車を表示するわけです。

車を画面の下、左端に表示したあとは、行番号100のFOR文の変数Iの初期値0のつぎの1から順に最終値の40までを、つぎつぎに行番号110のPUT@文のカッコのなかのIに送ってきますから、車はつぎの頁の実行結果に示すように、画面の下を左端から画面の中央まで移動していきます。

```

10 DEF USR=&HD000
130 A=USR(0)

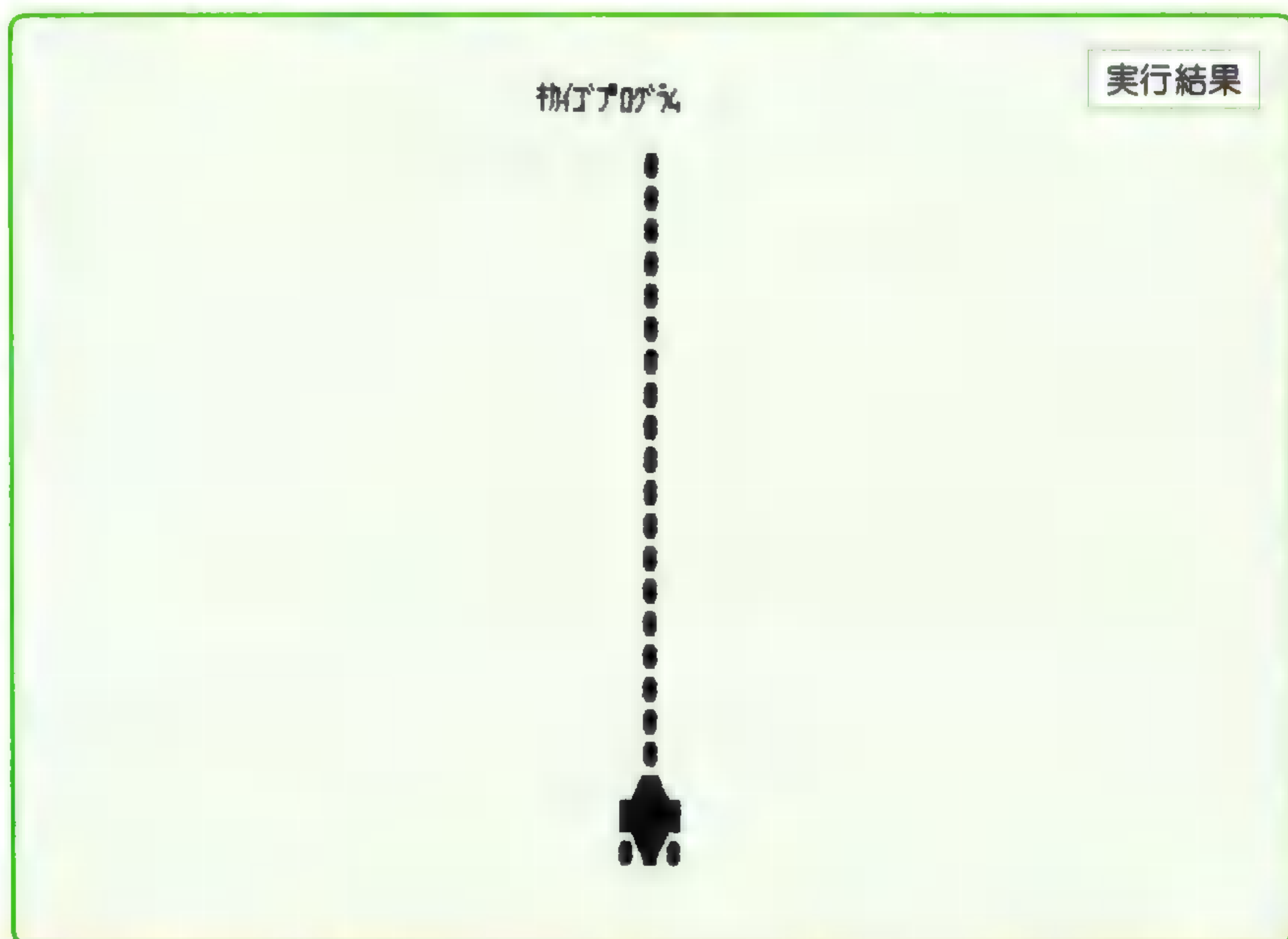
```


ベーシックとマシン語でできたプログラムをカセットテープに記録する

行番号130のA=USR (0) はマシン語のサブルーチンを呼び出します。

つまり、行番号10で、DEF USR=&HD000としてマシン語のプログラムが記憶されている最初のメモリ番地を指定していますから、行番号130のA=USR (0) を実行すると、D000番地へ実行が移ることになります。

さて、このマシン語のプログラムは、つぎの実行結果に示すように弾を画面のうえに向けて、音を出しながら発射し、弾が画面のうえに達したとき「キカイゴプログラム」というメッセージを表示するまでを行っています。



マシン語のプログラムを実行し終わると行番号140のLOCATE文を実行します。

```
140 LOCATE 0,15  
150 END
```

このLOCATE文は、行番号150のEND文を実行して表示するOKの表示位置を指定しています。このようにするのは、画面に表示された実行結果を壊さないためです。

ベーシックとマシン語でできたプログラムをパソコンのメモリに読み込む

ベーシックを読み込むにはCLOAD、マシン語はL

55頁で、ベーシックとマシン語でできたプログラムをカセットテープに記録しました。カセットテープに記録したベーシックとマシン語でできたプログラムは、必要に応じて、パソコンのメモリのなかに読み込んで使わなければなりません。そこでここでは、カセットテープに記録したベーシックとマシン語のプログラムを、カセットテープからパソコンのメモリへ読み込む方法について説明することになります。

読み込む方法といっても非常に簡単なことです。ベーシックのプログラムを、カセットテープからパソコンのメモリに読み込むには、

CLOAD “ファイル・ネーム”

を使って行います。

マシン語のプログラムをカセットテープからパソコンのメモリに読み込むには、50頁で説明したように、

L 

を使って行います。

ベーシックとマシン語でできたプログラムをカセットテープからパソコンのメモリに読み込むには、このふたつのコマンドを使って行えばよいのです。

では、実際にカセットテープに記録したベーシックとマシン語でできたプログラムを、パソコンのメモリに読み込んでみることにします。

ここでは、55頁で、CSAVEとWコマンドで、ベーシックとマシン語でできたプログラムをカセットテープに記録しましたから、そのプログラムをカセットテープから、パソコンのメモリに読み込むことにします。

まず、パソコンの電源を切り、パソコンのメモリに記憶されているものを消去します。

まず、ベーシックのプログラムを読み込む

55頁でベーシックとマシン語でできたプログラムをカセットテープに記録したとき、ベーシックのプログラムをさきにカセットテープに記録しましたから、ベーシックのプログラムをさきに、パソコンのメモリに読み込みます。

ベーシックのプログラムをカセットテープに記録するとき、ファイル・ネームを“a a”としました。したがって、

CLOAD “a a”

と入力します。そして、カセットテープを巻き戻すなり、先送りするなりして、“a a”の記録されている位置を合わせます。“a a”のある位置を合わせたら、データレコーダの**PLAY**ボタンを押して、**RET**キーを叩きます。

カセットテープのなかからファイル・ネーム“a a”を捜し出すと、

Found: aa

を表示して、ベーシックのプログラムの読み込みを開始します。読み込みが終了すると、**OK**を表示しますから、そこで、データレコーダの**STOP**ボタンを押して、データレコーダを止めます。

つぎに、マシン語のプログラムを読み込む

つぎは、マシン語のプログラムの読み込みですから、**MON**コマンドを入力して、マシン語のコマンド・レベルにします。

mon

* ← マシン語のコマンド・レベルにする

そして、*印の横に**L**コマンドを入力します。

mon

* **L** ← **L**コマンドを入力して、マシン語のプログラムを読み込む

Lコマンドを入力したら、データレコーダの**PLAY**ボタンを押し、**RET**キーを叩きます。すると、カーソルが**L**コマンドの横の*印のうえに移動して点滅し、マシン語のプログラムの読み込みを開始します。

mon

* **L** ← カーソルがこの位置で点滅する

マシン語のプログラムの読み込みが終了すると、つぎの行に*印が表示されてマシン語のコマンド・レベルに戻ります。

このプログラムはベーシックのプログラムからマシン語のプログラムを呼び出して実行しますから*印が表示されてマシン語のコマンド・レベルに戻ったら、**CTRL**キーと**B**のキーを一緒に押して、ベーシックのコマンド・レベルに戻します。そして、**RUN**とキー入力すれば、プログラムが実行されます。

つぎに、ベーシックとマシン語でできたプログラムの読み込みを行っている実行結果を示します。

```
NEC PC-8001 BASIC Ver 1.1  
Copyright 1979 (C) by Microsoft
```

実行結果

```
Ok  
cload"aa"  
Found:aa  
Ok  
mon  
*L  
*5  
Ok
```

←ベーシックのプログラムの読み込み

←マシン語のプログラムの読み込み

←**CTRL**キーと**B**のキーを一緒に押して
ベーシックのコマンド・レベルに戻す

```
h auto go to list run
```



パソコンのメモリの良否を チェックするには

メモリのチェックは、TMコマンド

パソコンのメモリが良いか悪いかを、チェックすることがあります。そのときに使うのが、**TMコマンド**です。

TMコマンドは、**TEST MEMORY (テスト・メモリ)** のことで、メモリが不良であるかないかのチェックを行います。

TMコマンドを使ってメモリのチェックを行うときは、つぎの実行結果に示すように、まず、**MON**コマンドを入力して、ベーシックのコマンド・レベルをマシン語のコマンド・レベルにします。

マシン語のコマンド・レベルになると、*****印が表示されますから、**TM**と入力します。そして、**[RET]** キーを叩きます。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok

mon

*TM

monコマンドを入力してマシン語の
コマンド・レベルにする

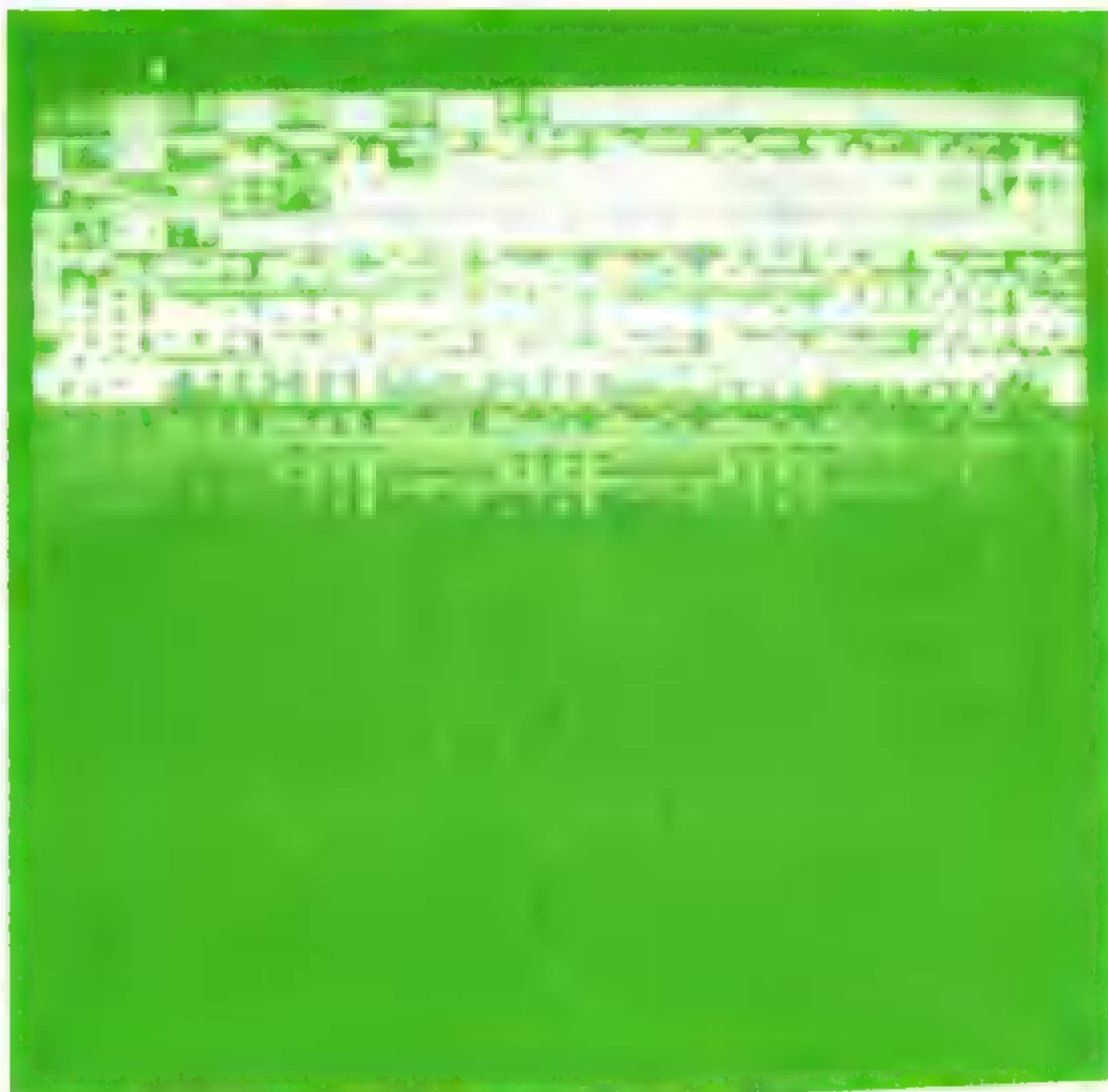
TMコマンドを入力して、**[RET]** キー
を叩く

auto auto list print

[RET] キーを叩くと、TMコマンドのまえにある*****印のうえにカーソルが移動して、そこで、カーソルが点滅します。

カーソルが*印のうえで点滅してしばらくすると、画面の一番下、つまり、ファンクションキーの内容が表示されている位置に、なにやら表示されて、画面が乱れ始めます。つぎに、画面の一番上のほうになにやら表示されて、画面が乱れます。この画面になにやら表示されて、画面が乱れるのは、メモリの不良からではなく、メモリ・テストが順調に行われているということです。

そしてさらに画面の乱れは、つぎの写真に示すように、しだいに広がっていきます。



画面の乱れが終わると、画面は、なにも表示されていない状態になります。ただカーソルだけは、もとの位置で点滅しています。

画面になにも表示されなくて、カーソルだけが点滅している状態は、32Kの場合、約3分ぐらいつづきますから、かなり長い時間のように感じられますが、約3分たつと、画面はパソコンに電源を入れた状態に戻ります。メモリ・テストに要する時間は、32Kの場合、約3分30秒です。

さて、画面がパソコンに電源を入れた最初の状態に戻ったときは、メモリ・テストの結果、メモリに異常がないということです。

▶メモリに不良があると、ブザーが鳴る

メモリに不良の箇所があるときは、不良が発見された時点でブザーが鳴りはじめます。ブザーは、**STOP** キーを叩いても鳴りつづけますから、このようなときは、電源を切るほかありません。

メモリ番地の読み方

8バイト表示のとき、メモリ番地は8つ置きに表示される

Dコマンドを使って、指定したメモリ番地のメモリに記憶されているマシン語を画面に表示させると、つぎの実行結果に示すように、各行の頭にメモリ番地が表示され、そのあとにマシン語が、1行に8個（8バイト）ずつ表示されていきます。

MON
*DD000,D01D

D000	06	12	3E	E9	21	00	F3	E5
D008	C5	06	28	77	23	23	10	FB
D010	C1	05	CA	1D	D0	E1	11	78
D018	00	19	C3	07	D0	76		

実行結果

← マシン語

← メモリ番地

Auto Run List

うえに示した実行結果は、メモリ番地D000からD01Dのメモリに記憶されているマシン語を、Dコマンドで画面に表示させた場合です。したがって、各行の頭にD000、D008、D010、D018というメモリ番地が表示されて、そのあとに8個（8バイト）ずつマシン語が表示されていきます。

さて、各行の先頭に表示されているメモリ番地は、どのメモリのメモリ番地を示しているのかというと、各行の最初に表示されているマシン語を記憶しているメモリのメモリ番地です。

つまり、D000は、1行目の最初の06を記憶しているメモリのメモリ番地、D008は、2行目の最初のマシン語C5を記憶しているメモリのメモリ番地、D010は、3行目の最初のマシン語C1を記憶しているメモリのメモリ番地、D018は、4行目の最初のマシン語00を記憶しているメモリのメモリ番地ということです。

▶ 表示されるのは最初のメモリ番地だけ

このように、各行の最初のマシン語が記憶されているメモリのメモリ番地は表示されますが、2番目からのマシン語が記憶されているメモリのメモリ番地は、ひとつひとつ表示されません。

したがって、2番目からのマシン語が記憶されているメモリのメモリ番地は、各行の先頭に表示されているメモリのメモリ番地にもとずいて、数えていくことになります。

この例の場合は、1行目に表示されているメモリ番地はD000で、そのメモリ番地は、最初のマシン語06を記憶しているメモリのメモリ番地ですから、2番目のマシン語12を記憶しているメモリのメモリ番地はD001、3番目の3Eを記憶しているメモリのメモリ番地はD002、4番目のマシン語E9を記憶しているメモリのメモリ番地はD003というように、D004、D005、D006、D007と、8個目のマシン語E5が記憶されているメモリのメモリ番地まで数えていき、2行目の先頭に移ります。

2行目の場合は、表示されているメモリのメモリ番地D008は、2行目の最初のマシン語C5を記憶しているメモリのメモリ番地ですから、2行目の2番目のマシン語06を記憶しているメモリのメモリ番地はD009、3番目のマシン語28を記憶しているメモリのメモリ番地はD00A、4番目のマシン語77を記憶しているメモリのメモリ番地はD00Bというように、D00C、D00D、D00E、D00Fと、8個目のマシン語FBが記憶されているメモリのメモリ番地まで数えていき、3行目の先頭に移ります。

▶ メモリ番地は、16進数でつけられている

この場合、D009のあと、D00A、D00B、D00C、D00D、D00E、D00Fと数えるのは、メモリのメモリ番地は16進数でつけられているからです。16進数については、91頁を参照してください。

▶ メモリ番地の読み方

つぎに、この場合のメモリ番地の読み方をまとめて示します。

	D000	D001	D002	D003	D004	D005	D006	D007
D000	06	12	3E	E9	21	00	F3	E5
	D008	D009	D00A	D00B	D00C	D00D	D00E	D00F
D008	C5	06	28	77	23	23	10	FB
	D010	D011	D012	D013	D014	D015	D016	D017
D010	C1	05	CA	1D	D0	E1	11	78
	D018	D019	D01A	D01B	D01C	D01D		
D018	00	19	C3	07	D0	76		

このように、1行にマシン語を8個（8バイト）ずつ表示させた場合は、メモリ番地は、8個置きに表示されます。

さて、画面にマシン語を表示させる場合は、普通は1行にマシン語を8個（8バイト）表示させます。なぜなら、1行8個（8バイト）のつぎは1行16個（16バイト）表示なので、文字が小さくなって読み取りづらいからです。

ただ、雑誌に掲載されているプログラムで、1行にマシン語が16個（16バイト）並んでいることがあります。この場合は、1行にマシン語が16個（16バイト）並びますから、メモリ番地は16個置きに表示されます。16個（16バイト）表示の場合のメモリ番地の読み方は、8個（8バイト）表示の場合のメモリ番地の読み方とおなじです。ただ、16個目で、つぎの行の先頭のメモリ番地に移るにすぎません。

	D000	D001	D002	D003	D004	D005	D006	D007
D000	06	12	3E	E9	21	00	F3	E5
	D008	D009	D00A	D00B	D00C	D00D	D00E	D00F
	C5	06	28	77	23	23	10	FB
	D010	D011	D012	D013	D014	D015	D016	D017
D010	C1	05	CA	1D	D0	E1	11	78
	D018	D019	D01A	D01B	D01C	D01D		
	00	19	C3	07	D0	76		

画面では、1行で表示される

チェックサムとは

チェックサムは、入力ミスがあるかないかを調べる

今日、雑誌などに、いろいろなマシン語のプログラムが掲載されています。雑誌などに掲載されているマシン語のプログラムを見ると、つぎのプログラムに示すように、プログラムの各行の終わりに「:」コロンで区切られて、そのあとに、マシン語とおなじ16進数が示されています。

D000	21	7B	FD	06	14	3E	EC	77	:	54
D008	16	FF	3E	20	D3	40	0E	FF	:	93
D010	0C	C2	10	D0	15	C2	0A	D0	:	5F
D018	3E	00	D3	40	3E	00	77	11	:	17
D020	78	00	ED	52	10	DF	06	0C	:	B8
D028	11	40	D0	21	9A	F3	1A	77	:	60
D030	23	13	10	FA	C9	FA	76	FF	:	78
D038	FF	FF	FF	FF	FF	FF	FF	FF	:	F8
D040	B7	B6	B2	BA	DE	CC	DF	DB	:	3D
D048	B8	DE	D7	D1	00	00	00	00	:	3E

チェックサム
(マシン語の
プログラムと
一緒に入力し
てはいけない)

この「:」コロンのあとにある16進数は、マシン語のプログラムではなくて、**Check sum (チェックサム)** というものです。**Check** (チェック) は、調べるとか照合する、という意味です。**sum** (サム) は、合計とか総和、という意味です。

このチェックサムは、私たちがマシン語のプログラムを入力したとき、果たして、入力したプログラムが間違いなく入力されているかどうかを確認するためにつけられています。

つまり、マシン語のプログラムを作った人が、そのプログラムを使う人がプログラムの入力を間違えないで入力することができるように、チェックサムをつけてくれているわけです。

チェックサムのつけ方

チェックサムのつけ方にはいろいろあります。前に取りあげたプログラムのチェックサムは、**単純サム方式**といい、各行のマシン語を合計していった、チェックサムとしたものです。

つまり、1行目の場合は、つぎに示すように、

$$D000 \quad 21 + 7B + FD + 06 + 14 + 3E + EC + 77 = 54$$

1行目の最初の21H（Hは、21が16進数であることを示す）から最後77Hまでを足していった、合計を出します。21Hから77Hまでを足すと354Hとなりますが、この354Hの下2桁54Hを、1行目のチェックサムとして、「:」コロンのあとにおきます。

2行目の場合は、2行目の最初の16Hから最後のFFHまでを足して、合計を出します。

$$D008 \quad 16 + FF + 3E + 20 + D3 + 40 + 0E + FF = 93$$

2行目の合計は393Hとなりますが、393Hの下2桁93Hを2行目のチェックサムとして、「:」コロンのあとにおきます。

3行目以降のチェックサムは、1行目、2行目の場合とおなじです。これが、チェックサムのつけ方です。

なお、チェックサムは、さきに説明したように、マシン語のプログラムが間違いなく入力されたかどうかを調べるためのものですから、マシン語のプログラムと一緒に入力してはなりません。

では、どのようにしてこのチェックサムで、入力したプログラムに入力ミスがあるかないかを調べるのか、チェックサムの使い方について説明します。

チェックサムの使い方

まずプログラムを入力します。ここでは、さきに示したマシン語のプログラムを入力します。プログラムを入力するメモリ番地はD000番地からです。

したがって、MONコマンドで、マシン語のプログラムが入力できるように、マシン語のコマンド・レベルにします。

mon

* ———マシン語のコマンド・レベルにする

マシン語のコマンド・レベルにしたら、*印のあとにSコマンドを入力し、

つづいてD000番地を入力します。

mon

***SD000** ← Sコマンドを入力して、D000番地を入力する

D000番地を入力したら、**RET** キーを叩きます。すると、

D000 FF- ← FF-に対して、マシン語のプログラムを入力する

D000番地を表示して、つづいてFF-を表示しますから、さきのプログラムを入力していきます。「:」コロンとチェックサムは入力しません。

さて、マシン語のプログラムを入力したら、入力したプログラムを画面に表示してみます。入力したマシン語のプログラムを画面に表示させるには、Dコマンドを使います。Dコマンドのあとには、マシン語を入力した最初のメモリ番地D000と最後のメモリ番地D04B番地を入力します。

***DD000, D04B** ← Dコマンドで入力したマシン語を画面に表示

すると、つぎのように入力したマシン語のプログラムが画面に表示されます。

mon

*DD000, D04B

D000 21 7B FD 06 14 3E EC 77

D008 16 FF 3E 20 D3 40 0E FF

D010 0C C2 10 D0 15 C2 0A D0

D018 3E 00 D3 40 3E 00 77 11

D020 78 00 ED 52 10 DF 06 0C

D028 11 40 D0 21 9A F3 1A 77

D030 23 13 10 FA C9 FA 76 FF

D038 FF FF FF FF FF FF FF FF

D040 B7 B6 B2 BA DE CC DF DB

D048 B8 DE D7 01

*■

実行結果

to auto go to list run

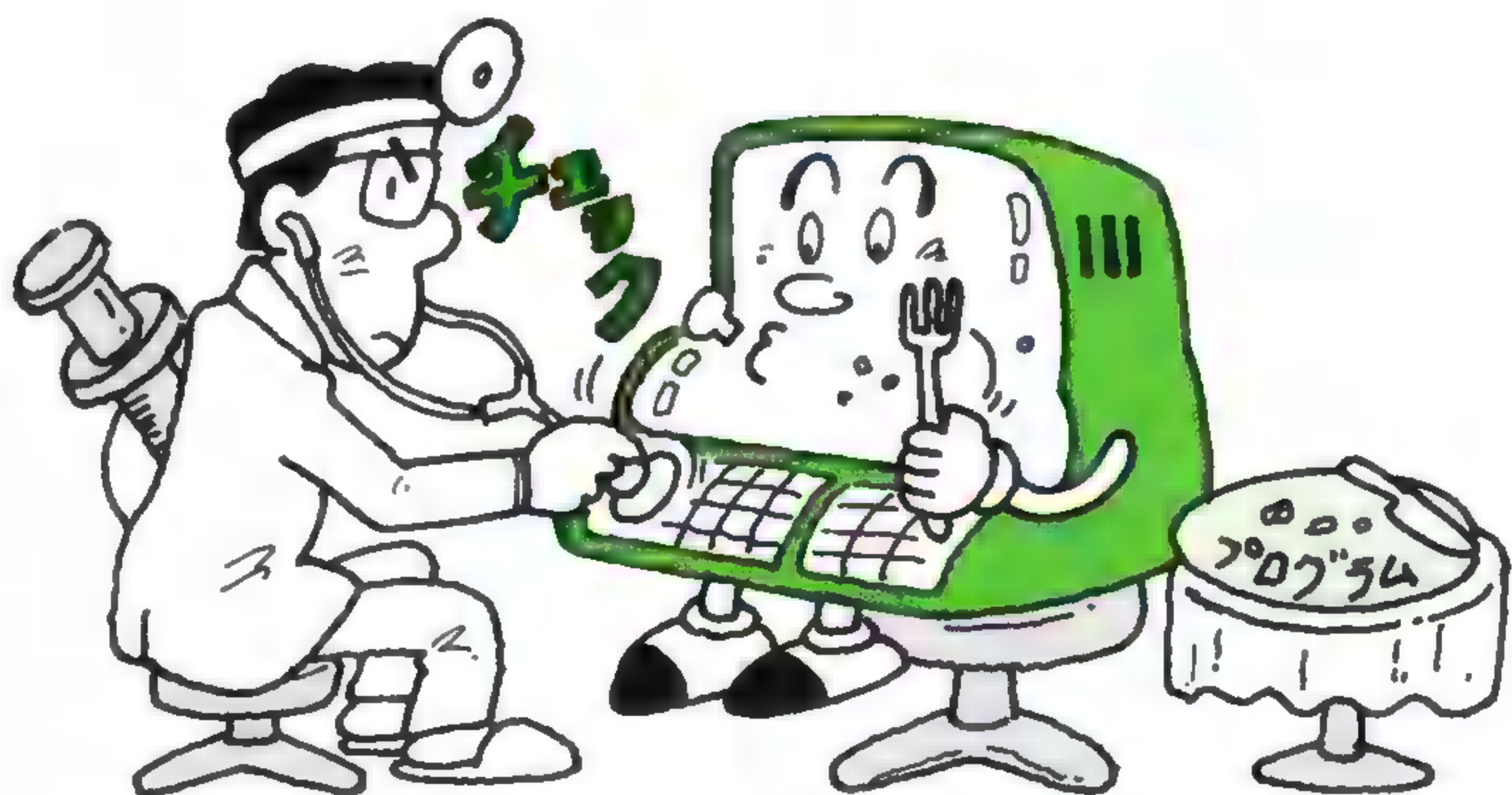
さて、画面に表示させたマシン語のプログラムが、間違いなく入力されているかどうかチェックしなければなりません。

短いマシン語のプログラムの場合ならまだしも、雑誌などに掲載されているような長いマシン語のプログラムを、間違いなく入力することはほとんど不可能に近いことだからです。マシン語のプログラムが長くなればなるほど、入力ミスの割合が高くなると考えて間違いありません。

入力したマシン語のプログラムに、入力ミスがあるかないか調べる場合、入力したマシン語のひとつひとつと、もとのマシン語のプログラムのひとつひとつを照らし合わせていたのでは大変です。そこで、入力したマシン語のチェックサムを求めます。そして、求めたチェックサムと、もとのマシン語のプログラムに示されているチェックサムとを照合します。入力したマシン語のチェックサムと、もとのプログラムに示されているチェックサムがおなじ数であると、入力したマシン語のプログラムには入力ミスがないということになります。逆にいえば、入力したマシン語のチェックサムと、もとのマシン語のプログラムに示されているチェックサムとが一致しないと、入力したマシン語に入力ミスがあるということになります。

これがチェックサムの考え方、あるいはチェックサムの役割というものです。

ただし、チェックサムは、パソコンにマシン語のプログラムを入力すると、自動的に算出されてつけられるものではありません。もとのプログラムのチェックサムであれ、入力したマシン語のチェックサムであれ、各自がチェックサムプログラムを作って求めています。このチェックサムの求め方にはいろいろな方法があって、もとのプログラムのチェックサムの求め方と、入力したマシン語のチェックサムの求め方が違うということもあります。このために、入力したマシン語のチェックサムと、もとのマシン語のプログラムのチェックサムとが、一致しないということもあるということを覚えておく必要があります。



入力したマシン語のチェックサムを求める

ではつぎに、入力したマシン語のチェックサムを求めることにします。まず、チェックサムを求めるプログラムを示します。

```
10 INPUT "スタートアドレス";ST$
20 INPUT "エンド アドレス";EN$
30 SA=VAL("&h"+ST$)
40 EA=VAL("&h"+EN$)
50 SN=0
60 FOR I=SA TO SA+7
70 SN=SN+PEEK(I)
80 NEXT
90 IF SA>=EA THEN END
100 PRINT HEX$(SA);" - ";HEX$(SA+7);"
    sum:";RIGHT$(HEX$(SN),2)
110 SA=SA+8
120 GOTO 50
```

このチェックサムを求めるプログラムは、さきのプログラムの場合とおなじように、各行の横1列の合計を求める単純サム方式です。

また、このチェックサムを求めるプログラムは、つぎの頁に示す実行結果のように、横1行の合計を算出して表示するプログラムで、さきに示したマシン語のプログラムのように、各行のマシン語のあとに、算出したチェックサムを表示するというものではありません。

チェックサム・プログラムの入力

さて、チェックサムを求めるプログラムの説明はあとで行うことにして、まず、チェックサムを求めるプログラムを入力して、実行させてみます。

さきに入力したマシン語のプログラムは、そのままにしておきます。

チェックサムを求めるプログラムは、ベーシックで書かれていますから、さきのマシン語のプログラムを入力し終った状態のままの場合は、**CTRL** キーと **B** のキーを一緒に押して、ベーシックのコマンド・レベルに戻します。そし

て、チェックサムを求めるプログラムを入力します。

ベーシックで書かれたプログラムは、マシン語のプログラムと違って自動的に8021番地のメモリから入力されていきますから、入力したチェックサムを求めるプログラムは、8021番地のメモリから記憶されていくことになります。したがって、D000番地から入力したマシン語のプログラムは、そのままD000番地から記憶されています。

チェックサムを求めるプログラムを入力し終わったら、RUNを入力します。すると、つぎのように、スタートアドレスの入力を求めてきますから、

run

スタートアドレス? d000

入力したマシン語のプログラムが記憶されている最初のメモリ番地D000を入力します。D000を入力してRETキーを叩くと、つぎにエンドアドレスを求めてきます。

run

スタートアドレス? d000

エンドアドレス? d04b

そこで、入力したマシン語のプログラムが記憶されている、最後のメモリ番地D04Bを入力します。そして、RETキーを叩くと、つぎの実行結果のように、入力したマシン語の各行のチェックサムが算出されて表示されます。

run
スタートアドレス? d000
エンド アドレス? d04b
D000 - D007
D008 - D00F
D010 - D017
D018 - D01F
D020 - D027
D028 - D02F
D030 - D037
D038 - D03F
D040 - D047
D048 - D04F

メモリ番地

sum:54
sum:93
sum:5F
sum:17
sum:B8
sum:60
sum:78
sum:F8
sum:3D
sum:3E

チェックサム

実行結果

Auto Stop Next Run

このようにして求めた、入力したマシン語のチェックサムと、もとのマシン語のプログラムに示されているチェックサムとを照合するわけです。

この場合、もとのマシン語のプログラムは、68頁に示したマシン語のプログラムです。したがって、そのプログラムに示されているチェックサムと照合することになります。うえから順に1行ずつチェックサムを照合していきます。この場合は、もとのプログラムのチェックサムと違うチェックサムがありませんから、入力ミスがないということになります。

▶入力ミスがある場合

では、つぎに入力ミスがある場合を示します。入力するプログラムはさきの場合とおなじです。

```
run
スタートアドレス? d000
エンド アドレス? d04b
D000 - D007      sum: 54
D008 - D00F      sum: 93
D010 - D017      sum: 5F
D018 - D01F      sum: 08
D020 - D027      sum: B8
D028 - D02F      sum: 60
D030 - D037      sum: 78
D038 - D03F      sum: F8
D040 - D047      sum: 3D
D048 - D04F      sum: 3E
Ok
```

実行結果

h 2010 2010 1111 1010

うえに示した入力したマシン語のチェックサムと、68頁に示したもとのプログラムのチェックサムとを、うえから順に1行ずつ照合していくと、4行目のチェックサムは08、もとのプログラムのチェックサムは17で、チェックサムが違っています。したがって、入力ミスがあるということになります。

そこで、MONコマンドを入力して、マシン語のコマンド・レベルにし、

mon

* ← MONコマンドで、マシン語のコマンド・レベルにする

* 印の横にDコマンドと、4行目のメモリ番地を入力して、

mon

***DD018, D01F** ← チェックサムが違っている行のメモリ番地を入力する

そのメモリ番地に記憶されているマシン語を、つぎの実行結果に示すように画面に表示します。そして、もとのプログラムのマシン語と、入力したマシン語をひとつひとつ照合します。この場合は、左から3個目のD3がC4と入力されていますから、このC4をD3に変更します。

入力ミスの変更の仕方については、31頁で説明していますから、31頁を参照してください。

mon

*DD018, D01F

D018 3E 00 C4 40 3E 00 77 11

*SD01A

D01A C4-D3 40-

チェックサムが違っているメモリ番地を入力する

実行結果

そのメモリ番地に記憶されているマシン語を表示

Sコマンドと違っているマシン語のメモリ番地を入力して、マシン語を変更する

Auto

Go to

Test

Print

チェックサム・プログラムの説明

では、入力したマシン語のチェックサムを求めるプログラムの、各ステートメントの働きについて、簡単に説明しておくことにします。

10 INPUT "スタートアドレス"; ST\$
20 INPUT "エンド アドレス"; EN\$

行番号10と20のINPUT文で、マシン語を入力した最初のメモリ番地と、最後のメモリ番地を入力します。行番号10のINPUT文でST\$に入力された最初のメモリ番地は、行番号30のVAL関数のST\$に送られます。行番号20のINPUT文でEN\$に入力された最後のメモリ番地は、行番号40のVAL関数のEN\$に送られます。


```
30 SA=VAL("&h"+ST$)
```

```
40 EA=VAL("&h"+EN$)
```

ST\$とEN\$に送られてきた最初のメモリ番地と最後のメモリ番地は、&Hがプラスされて、VAL関数で16進数の数値に変換されます。そして、=の左側にあるSAとEAにそれぞれ代入されます。

行番号50で、SNに0を代入します。この理由は、あとでわかります。

```
50 SN=0
```

行番号60のFOR文と行番号80のNEXT文で、初期値SAから最終値SA+7まで、その間にある行番号70のステートメントを繰り返します。

```
60 FOR I=SA TO SA+7
```

```
80 NEXT
```

FOR文のSAには、最初、行番号30のVAL関数で16進数の数値に変換されて、SAに代入された値D000が送られてきますから、FOR文は、

```
FOR I=D000 TO D000+7
```

となります。FOR文の変数Iの初期値D000から最終値D000+7は、行番号70のPEEK関数のIに送られ、PEEK関数でマシン語を取り出すメモリ番地の指定となります。

```
70 SN=SN+PEEK(I)
```

PEEK関数のIに送られてくる16進数はD000ですから、

```
PEEK (D000)
```

となって、D000番地に記憶されている21Hを取り出してきて、SNの値と足し算されます。最初SNは0ですから、00H+21Hが行われます。足し算の結果の21Hは、=の左側のSNに代入されて記憶されます。

つぎにFOR文の変数Iの値はD001Hとなりますから、その値をPEEK関数のIに送ります。

```
PEEK (D001)
```

そして、PEEK関数で、D001番地に記憶されている7BHを取り出してきて、SNに代入した21Hと足します。21H+7BHの結果の9CがSN

に代入されて、SNは9Cとなります。

このことをFOR文の変数Iの初期値D000から最終値D000+7まで繰り返えすと、PEEK関数によって、メモリ番地D000からD007までに記憶されているマシン語が取り出されてきて足し算されることになります。

FOR文の変数Iの値が最終値まで繰り返えすと、FOR文の繰り返えしをやめて、行番号90のIF文を実行します。

```
90 IF SA>=EA THEN END
```

行番号90のIF文は、行番号20で入力した最後のメモリ番地になったかどうかを判定しています。最後のメモリ番地になると、SA>=EAの条件が成立して、THEN ENDを実行して、終わりとなります。

この場合は、まだ1回目ですから、IF文の条件は成立しません。したがって、行番号100のPRINT文の実行に進みます。

```
100 PRINT HEX$(SA); " - "; HEX$(SA+7); "  
sum: "; RIGHT$(HEX$(SN), 2)
```

行番号100のPRINT文で、メモリ番地とその番地に記憶されているマシン語の合計結果（チェックサム）を表示します。

HEX\$関数は、SAの値を文字型の16進数に変換する関数で、最初は、

```
PRINT HEX$(SA); " - "; HEX$(SA+7)
```

で、つぎのように表示されます。

```
D000 - D007
```

RIGHT\$関数は、文字列の右から指定しただけの文字を取り出す関数です。

```
RIGHT$(HEX$(SN), 2)
```

この場合はHEX\$関数で文字型の16進数に変換したものから、右から2文字取り出すことになります。取り出した2文字は、PRINT文で表示されます。最初のD000番地からD007番地までの合計の結果は354Hです。したがって右から2文字取り出すと54Hになりますから、つぎのように表示されます。

```
sum: 54
```

メモリ番地と合計の結果を表示すると、行番号110のステートメントを実行します。

```
110 SA=SA+8
```


行番号110の $SA = SA + 8$ で、これまでの SA の値に 8 を足します。その結果の値を = の左側の SA に代入します。

そして、行番号120のGOTO文で、行番号50に戻ります。

120 GOTO 50

行番号50の $SN = 0$ で SN に 0 を代入して、 SN に記憶されている D000 番地から D007 番地までのマシン語の合計結果を 0 に戻します。

SN を 0 に戻すと、行番号60のFOR文と行番号80のNEXT文で、行番号70の $SN = SN + PEEK(I)$ を、FOR文の変数 I の初期値 SA から最終値 $SA + 7$ まで繰り返えします。こんどは、行番号110で SA の値に 8 を足していますから、 SA は D008 となっています。したがって、

FOR I=D008 TO D008+7

となります。このFOR文の変数 I の初期値 D008 から最終値 D00F までを PEEK 関数の I に送って、PEEK 関数で D008 番地から D00F 番地に記憶されているマシン語を取り出します。そして、前回とおなじように足していった合計を算出します。これまで説明したことを繰り返えして、行番号10と20のINPUT文で入力した、最初のメモリ番地から最後のメモリ番地までの合計（チェックサム）を算出していきます。

以上説明したのは、1行のマシン語が8個（8バイト）の場合のチェックサムですが、1行のマシン語が16個（16バイト）の場合のチェックサムもあります。1行16バイトの場合は、さきに示したチェックサム・プログラムを、つぎに示すように変更して使用してください。つぎに、さきに示したチェックサム・プログラムの変更する行番号60から行番号110までを示します。

```
60 FOR I=SA TO SA+15 * 8を15に変更する
70 SN=SN+PEEK(I)
80 NEXT
90 IF SA>=EA THEN END
100 PRINT HEX$(SA); " - "; HEX$(SA+15); "
    sum: "; RIGHT$(HEX$(SN), 2)
110 SA=SA+16 * 8を16に変更する
```


ビットとバイト

1 ビットは2進数の1桁を表わす

マシン語の場合、**ビット**とか**バイト**とかいう言葉がさかんに出てきます。まず、ビットという言葉から説明します。**ビット (bit)** とは、バイナリー・ディジット (binary digit) の略で、情報量の単位を表わします。バイナリーとは、「2進法の」という意味で、ディジットは「数字」という意味です。

1 ビットという場合は、つぎに示すように、2進数の1桁のことをいいます。

1 ← 1 ビット

0 ← 1 ビット

2 ビットという場合は、つぎに示すように、2進数の2桁のことをいいます。

00
01
10
11

2 ビット

4 ビットという場合は、つぎに示すように、2進数の4桁のことをいいます。

4 ビットのことを1 ニブルということもあります。

0000
0001
0010
0011
0100

4 ビット

さて、8 ビットという場合は、つぎに示すように、2進数の8桁のことをいいます。

00111110
11101001
00110010
11100100

8 ビット

8ビットは1バイト

8ビットは、1バイトとともいいます。**バイト (byte)** は8ビットをひとつの単位として表わす表わし方です。つまり、12個をひとつの単位として、1ダースといういい表わし方とおなじです。

8ビットをわざわざ1バイトといいかえるのは、ほとんどのCPUが、8ビットを1単位として、情報を扱っているからです。

ひとつのメモリも8ビットを記憶する

CPUは情報を8ビット、つまり1バイトを1単位として扱っています。したがって、ひとつひとつのメモリも8ビット、つまり、1バイトずつ情報を記憶するようになっています。

また、メモリの容量の大きさを表現するときも、このバイトという単位を使います。たとえば、16Kバイトとか、32Kバイトとかいうようにです。

16進数1桁は4ビット

91頁で16進数について説明していますが、4ビットの2進数で表わすことができる0000から1111までは、16進数の0からFまでの1桁で表わします。

したがって、16進数の0からFまでの1桁は、4ビットとなります。

0 1 2 3 4 5 6 7 8 9 A B C D E F

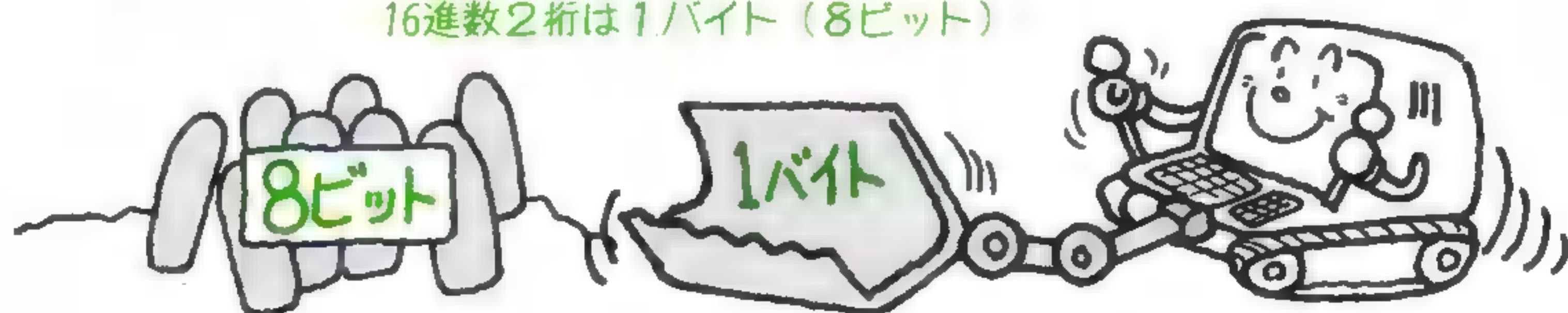
16進数1桁は4ビット

16進数の2桁は1バイト

また、8ビットの2進数で表わすことができる00000000から11111111までは16進数の00からFFで表わされます。したがって、16進数の00からFFまでの2桁は、8ビットということになります。8ビットは1バイトですから、16進数2桁を1バイトといいます。

00 1A 1F 10 2C 37 AA D5 E0

16進数2桁は1バイト(8ビット)



2進数は、このように書き表わされている

2進数は、1、2、4、8の値に電気をつけて数を表わす

まず、2進数について考えるとき、私たちがふだん数を表わすために使っている10進数については、考えないようにしなければなりません。なぜなら、10進数が頭のなかにあると、常に2進数を10進数に置き換えてとらえてしまうからです。2進数を10進数に置き換えてとらえてしまっていては、いつまでたっても2進数はわからないものになってしまいます。

さてパソコンは、電気回路でできています。電気回路でできているパソコンは、電気信号であらゆることを処理しているわけですが、この電気信号には、電気回路がつながっているか、いないかの2つしかありません。つまりオンとオフの2つの状態です。このように、パソコンにはオンとオフの2つしかないのです。このオンとオフの2つで数を表わすところから、2進数で数を表わすということになっているのです。

では、このオンとオフのふたつで、どうやって数を表わすのでしょうか。

ただオンとオフだけでは、数を表わすことはできません。うまいことに、2進数には、1、2、4、8という数のまとまりがあります。そこで、つぎに示すように、ケイを引いてメモリをきざみます。そして、そのメモリに、1、2、4、8という値をふります。

+	-	-	-	-	+	-	-	-	-	+	-	-	-	-	+
8					4					2					1

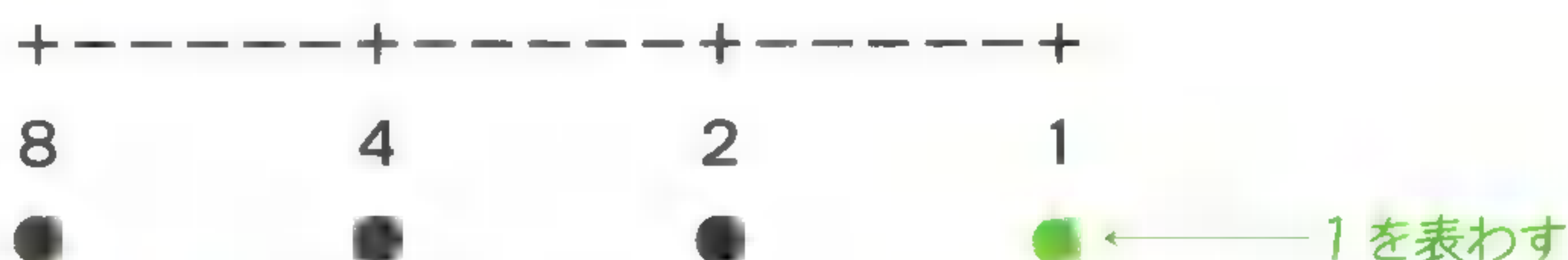
ここでは、8までとします。あとで、8以降の値についても説明します。さて、オンは電気がついた状態ですから、●で示します。オフは電気がついていない状態ですから、●で示します。

0という数は、つぎに示すように、どの値にも電気がついていない状態です。

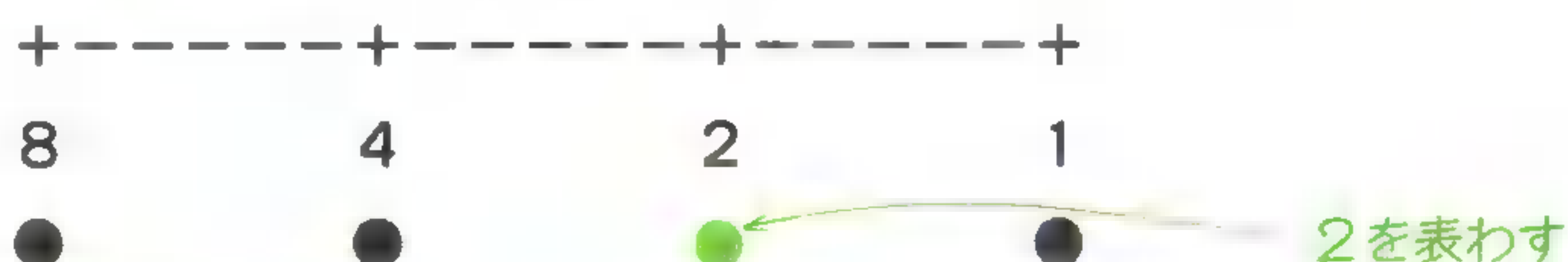
+	-	-	-	-	+	-	-	-	-	+	-	-	-	-	+
8					4					2					1
●					●					●					●

← 0を表わす

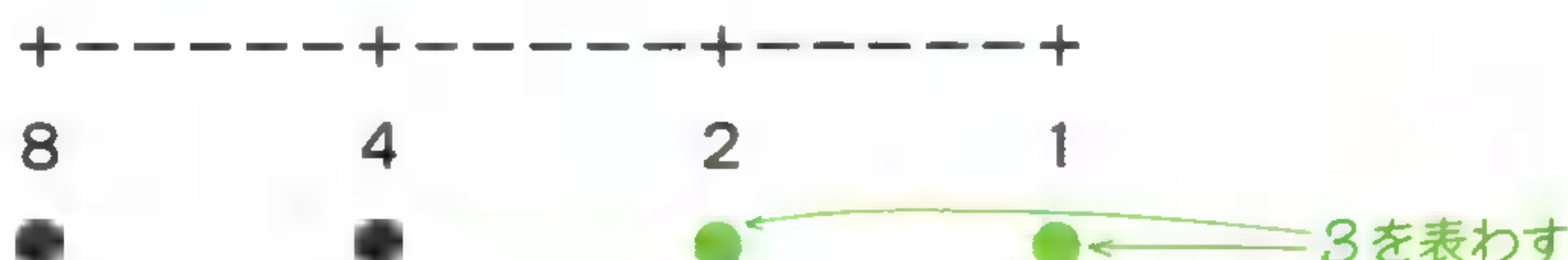
1 という数は、1 という値がメモリにふられていますから、1 の値のところに電気をつけます。



2 という数は、2 という値がメモリにふられていますから、2 の値のところに電気をつけます。

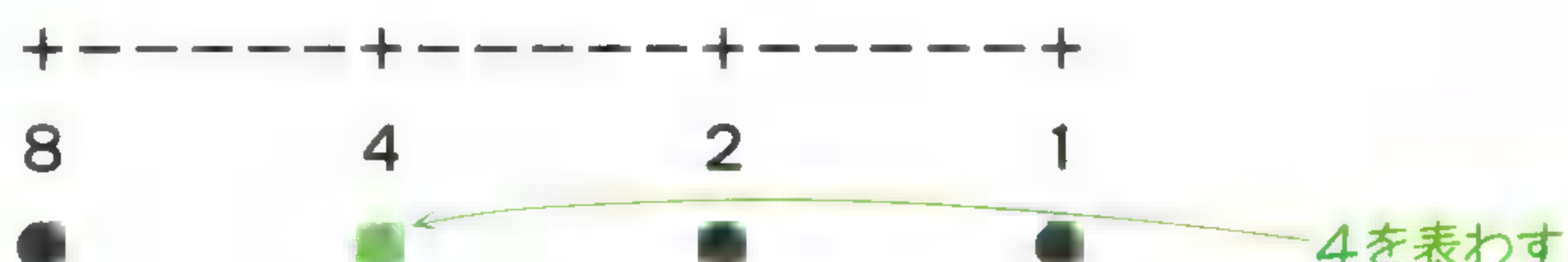


つぎは**3 という数**です。3 という値は、メモリにふられていません。では、どうやって3 という数を表わすのでしょうか。2 の値と1 の値のところに電気をつけて表わします。

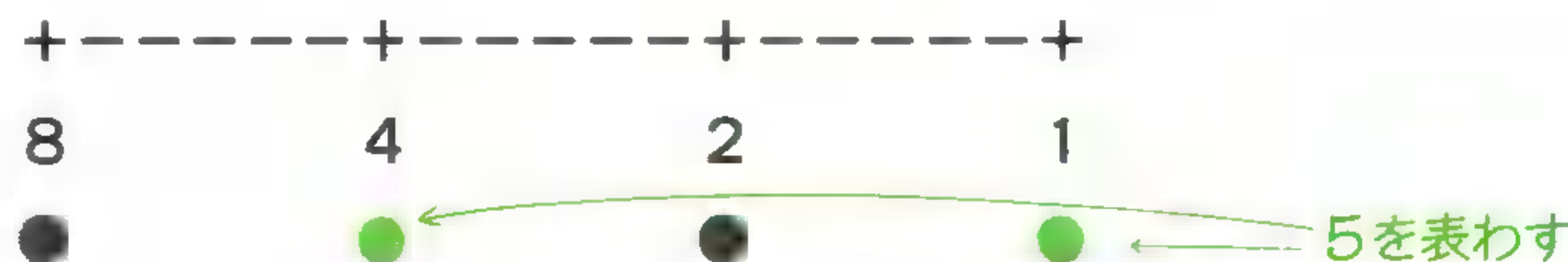


1 の値と2 の値に電気がつくということは、1 の値と2 の値を足すことを示しています。1 + 2 は3 ですから、1 の値と2 の値に電気がつくことは、3 を示していることになるのです。このことは、2 進数を10 進数に直したり、10 進数を2 進数に直したりするときを使うので、よくおぼえておくことです。

4 の数は、メモリに4 の値がふられていますから、4 の値のところに電気をつけます。

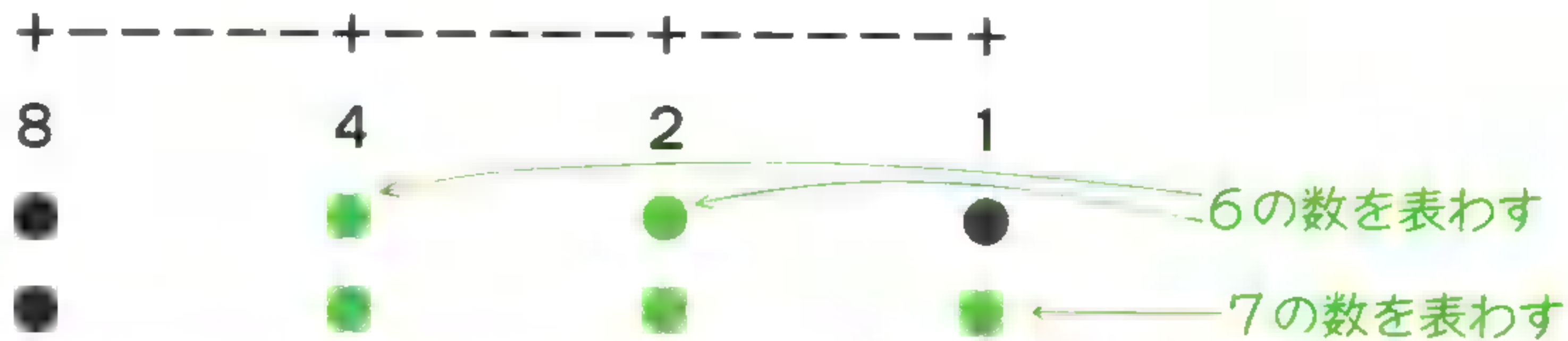


5 の数は、メモリに5 の値がふられていません。したがって、3 の数を表わしたときのように、メモリにふられている値から、5 の数になる値を選び出して、その値に電気をつけることになります。1、2、4、8 の値のうち、5 の数になる値は1 と4 ですから、1 と4 の値のところに電気をつけます。



2進数は、このように書き表わされている

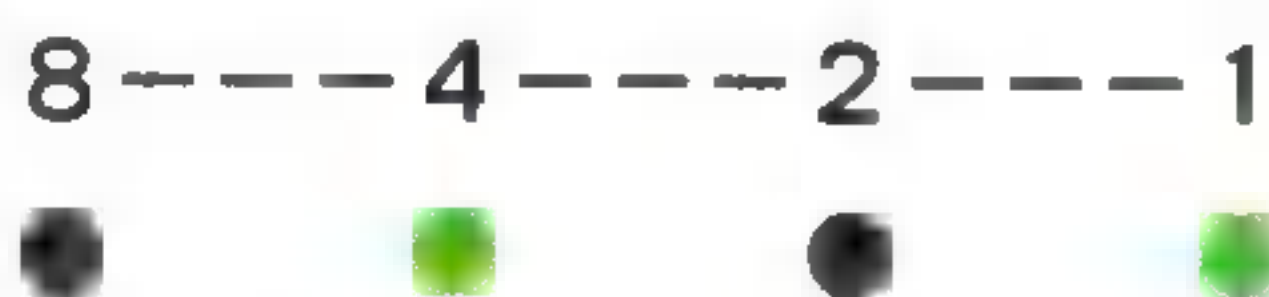
6の数と7の数は、メモリに6の値、7の値がふられていませんから、5の数とおなじように、1、2、4、8のなかから、6の数、7の数になる値を選び出して、その値に電気をつけます。6の数は2の値と4の値、7の数は1の値と2の値と4の値に電気をつければよいわけですから、つぎのように電気をつけます。



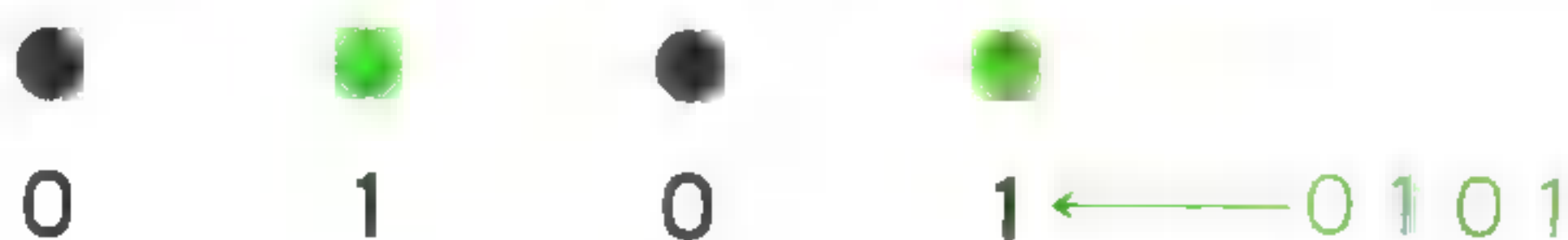
オンに1、オフに0をあてはめる

ここまで、メモリにふられている値に、表わす数に応じて電気をつけてきました。ただ、電気をつけただけでは、私たちが数として表わすには不便です。そこで、オンに1、オフに0をあてはめて、1と0で表わす2進数としているわけです。したがって、これまで説明してきた電気がついているところに1、電気がついていないところに0をそのままあてはめると、2進数と呼んでいる1と0で表わされた数そのものになります。

まず、5を1と0の2進数で表わしてみます。5はつぎに示すように、8と2の値のところが電気がつかず、4と1の値のところに電気がつきました。

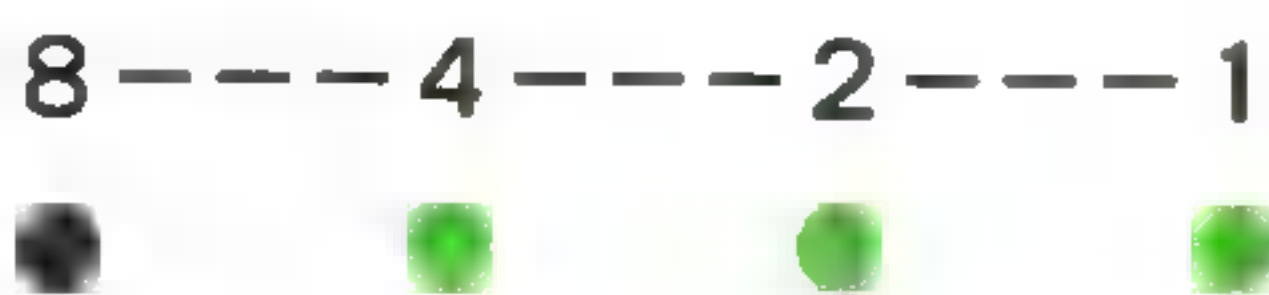


電気がついたところに1、電気がつかなかったところに0をあてはめます。



このように5は、2進数で0101で表わされるということになります。

つぎに7を2進数で表わしてみます。7は、1と2と4に電気がつきました。



電気がついたところに1、電気がつかないところに0をあてはめます。



7は、2進数で0111ということになります。

これが2進数の表わし方です。つぎに10進数と2進数の対応表を示しますから、さきに説明したように、メモリに1、2、4、8の値をふって、それぞれの2進数の1と0をその値のところにおいてみてください。そうすることによって、1と0の並びが、なぜ10あるいは15などという数を示すのか、よりよく理解できるようになると思います。

▶ 10進数、2進数対応表

10進数	2進数	8	4	2	1
0	0000	●	●	●	●
1	0001	●	●	●	●
2	0010	●	●	●	●
3	0011	●	●	●	●
4	0100	●	●	●	●
5	0101	●	●	●	●
6	0110	●	●	●	●
7	0111	●	●	●	●
8	1000	●	●	●	●
9	1001	●	●	●	●
10	1010	●	●	●	●
11	1011	●	●	●	●
12	1100	●	●	●	●
13	1101	●	●	●	●
14	1110	●	●	●	●
15	1111	●	●	●	●

8ビットの場合は

これまで説明してきた2進数は、4ビットで表わすことができる15までの数です。ビットについては79頁で説明していますから参照してください。

私たちがふつう使っているパソコンは、8ビット型パソコンですから、当然4ビットで表わすことができる15以上の数を表わすことができます。

8ビットの場合は、つぎに示すように、4ビットの場合のメモリに、さらに4つメモリが増えて、そのメモリに16、32、64、128の値をふります。

+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+
128				64				32				16				8				4			2				1					

電気のつき方は、4ビットの2進数で説明したこととおなじです。つまり、表わす数に応じた値を、メモリにふられている値のなかから選び出して、その値のところに電気がつくことになります。

まず、16の数から始めます。16の数は、メモリに16という値がふられていますから、16の値に電気がつきます。そのほかには電気がつきませんから、つぎのようになります。その下に、電気がついたところに1、電気につかないところに0をあてはめて、2進数を示します。

●●●●●●●●
00010000

つぎは、17の数です。17という数は、メモリにふられていませんから、メモリにふられている値から、17という数になる値を選び出して、その値に電気がつきます。17の場合は、1の値と16の値とで17になりますから、1の値と16の値のところに電気がつきます。そのほかには、電気につかないので、つぎのようになります。

●●●●●●●●
00010001

18の数も、メモリにふられていません。18の場合は、2の値と16の値とで18になりますから、2の値と16の値のところに電気がつきます。そのほかには、電気につかないので、つぎのようになります。

●●●●●●●●
00010010

19の数も、おなじようにメモリにふられていません。19の場合は、1の値と2の値、それに16の値とで19になりますから、1と2と16の値のところに電気がつきます。そのほかのところにはつきませんから、つぎのようになります。

●●●●●●●●
00010011

あとは、つぎに10進数と8ビットの2進数の対応表を示しますので、さきに示したように、メモリに1、2、4、8、16、32、64、128の値をふって、それ

それぞれの2進数の1と0を、その値のところにおいてみて、どのように表わされているか確かめてみてください。

なお、8ビットで表わすことができる数は、全部の値に電気がついた11111111（255）までです。

10進数と8ビットの2進数対応表

10進数	2進数	10進数	2進数
0	00000000	24	00011000
1	00000001	25	00011001
2	00000010	26	00011010
3	00000011	27	00011011
4	00000100	28	00011100
5	00000101	29	00011101
6	00000110	30	00011110
7	00000111	31	00011111
8	00001000	32	00100000
9	00001001	33	00100001
10	00001010	34	00100010
11	00001011	35	00100011
12	00001100	36	00100100
13	00001101	37	00100101
14	00001110	38	00100110
15	00001111	39	00100111
16	00010000	40	00101000
17	00010001	41	00101001
18	00010010	42	00101010
19	00010011	43	00101011
20	00010100	44	00101100
21	00010101	45	00101101
22	00010110	46	00101110
23	00010111	47	00101111

2進数は、このように書き表わされている

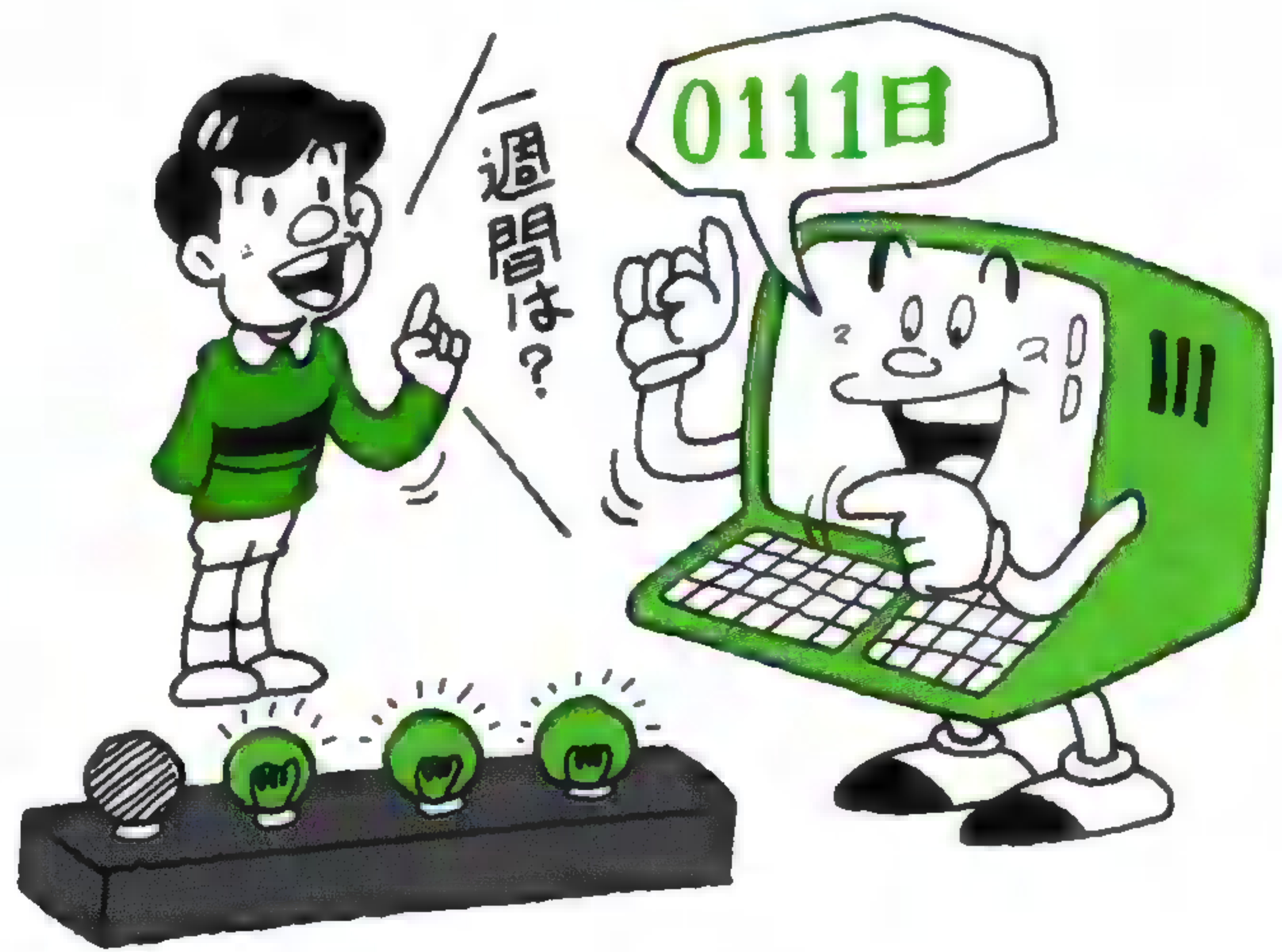
48	00110000	79	01001111
49	00110001	80	01010000
50	00110010	81	01010001
51	00110011	82	01010010
52	00110100	83	01010011
53	00110101	84	01010100
54	00110110	85	01010101
55	00110111	86	01010110
56	00111000	87	01010111
57	00111001	88	01011000
58	00111010	89	01011001
59	00111011	90	01011010
60	00111100	91	01011011
61	00111101	92	01011100
62	00111110	93	01011101
63	00111111	94	01011110
64	01000000	95	01011111
65	01000001	96	01100000
66	01000010	97	01100001
67	01000011	98	01100010
68	01000100	99	01100011
69	01000101	100	01100100
70	01000110	101	01100101
71	01000111	102	01100110
72	01001000	103	01100111
73	01001001	104	01101000
74	01001010	105	01101001
75	01001011	106	01101010
76	01001100	107	01101011
77	01001101	108	01101100
78	01001110	109	01101101

110	0 1 1 0 1 1 1 0	141	1 0 0 0 1 1 0 1
111	0 1 1 0 1 1 1 1	142	1 0 0 0 1 1 1 0
112	0 1 1 1 0 0 0 0	143	1 0 0 0 1 1 1 1
113	0 1 1 1 0 0 0 1	144	1 0 0 1 0 0 0 0
114	0 1 1 1 0 0 1 0	145	1 0 0 1 0 0 0 1
115	0 1 1 1 0 0 1 1	146	1 0 0 1 0 0 1 0
116	0 1 1 1 0 1 0 0	147	1 0 0 1 0 0 1 1
117	0 1 1 1 0 1 0 1	148	1 0 0 1 0 1 0 0
118	0 1 1 1 0 1 1 0	149	1 0 0 1 0 1 0 1
119	0 1 1 1 0 1 1 1	150	1 0 0 1 0 1 1 0
120	0 1 1 1 1 0 0 0	151	1 0 0 1 0 1 1 1
121	0 1 1 1 1 0 0 1	152	1 0 0 1 1 0 0 0
122	0 1 1 1 1 0 1 0	153	1 0 0 1 1 0 0 1
123	0 1 1 1 1 0 1 1	154	1 0 0 1 1 0 1 0
124	0 1 1 1 1 1 0 0	155	1 0 0 1 1 0 1 1
125	0 1 1 1 1 1 0 1	156	1 0 0 1 1 1 0 0
126	0 1 1 1 1 1 1 0	157	1 0 0 1 1 1 0 1
127	0 1 1 1 1 1 1 1	158	1 0 0 1 1 1 1 0
128	1 0 0 0 0 0 0 0	159	1 0 0 1 1 1 1 1
129	1 0 0 0 0 0 0 1	160	1 0 1 0 0 0 0 0
130	1 0 0 0 0 0 1 0	161	1 0 1 0 0 0 0 1
131	1 0 0 0 0 0 1 1	162	1 0 1 0 0 0 1 0
132	1 0 0 0 0 1 0 0	163	1 0 1 0 0 0 1 1
133	1 0 0 0 0 1 0 1	164	1 0 1 0 0 1 0 0
134	1 0 0 0 0 1 1 0	165	1 0 1 0 0 1 0 1
135	1 0 0 0 0 1 1 1	166	1 0 1 0 0 1 1 0
136	1 0 0 0 1 0 0 0	167	1 0 1 0 0 1 1 1
137	1 0 0 0 1 0 0 1	168	1 0 1 0 1 0 0 0
138	1 0 0 0 1 0 1 0	169	1 0 1 0 1 0 0 1
139	1 0 0 0 1 0 1 1	170	1 0 1 0 1 0 1 0
140	1 0 0 0 1 1 0 0	171	1 0 1 0 1 0 1 1

2進数は、このように書き表わされている

172	1 0 1 0 1 1 0 0	203	1 1 0 0 1 0 1 1
173	1 0 1 0 1 1 0 1	204	1 1 0 0 1 1 0 0
174	1 0 1 0 1 1 1 0	205	1 1 0 0 1 1 0 1
175	1 0 1 0 1 1 1 1	206	1 1 0 0 1 1 1 0
176	1 0 1 1 0 0 0 0	207	1 1 0 0 1 1 1 1
177	1 0 1 1 0 0 0 1	208	1 1 0 1 0 0 0 0
178	1 0 1 1 0 0 1 0	209	1 1 0 1 0 0 0 1
179	1 0 1 1 0 0 1 1	210	1 1 0 1 0 0 1 0
180	1 0 1 1 0 1 0 0	211	1 1 0 1 0 0 1 1
181	1 0 1 1 0 1 0 1	212	1 1 0 1 0 1 0 0
182	1 0 1 1 0 1 1 0	213	1 1 0 1 0 1 0 1
183	1 0 1 1 0 1 1 1	214	1 1 0 1 0 1 1 0
184	1 0 1 1 1 0 0 0	215	1 1 0 1 0 1 1 1
185	1 0 1 1 1 0 0 1	216	1 1 0 1 1 0 0 0
186	1 0 1 1 1 0 1 0	217	1 1 0 1 1 0 0 1
187	1 0 1 1 1 0 1 1	218	1 1 0 1 1 0 1 0
188	1 0 1 1 1 1 0 0	219	1 1 0 1 1 0 1 1
189	1 0 1 1 1 1 0 1	220	1 1 0 1 1 1 0 0
190	1 0 1 1 1 1 1 0	221	1 1 0 1 1 1 0 1
191	1 0 1 1 1 1 1 1	222	1 1 0 1 1 1 1 0
192	1 1 0 0 0 0 0 0	223	1 1 0 1 1 1 1 1
193	1 1 0 0 0 0 0 1	224	1 1 1 0 0 0 0 0
194	1 1 0 0 0 0 1 0	225	1 1 1 0 0 0 0 1
195	1 1 0 0 0 0 1 1	226	1 1 1 0 0 0 1 0
196	1 1 0 0 0 1 0 0	227	1 1 1 0 0 0 1 1
197	1 1 0 0 0 1 0 1	228	1 1 1 0 0 1 0 0
198	1 1 0 0 0 1 1 0	229	1 1 1 0 0 1 0 1
199	1 1 0 0 0 1 1 1	230	1 1 1 0 0 1 1 0
200	1 1 0 0 1 0 0 0	231	1 1 1 0 0 1 1 1
201	1 1 0 0 1 0 0 1	232	1 1 1 0 1 0 0 0
202	1 1 0 0 1 0 1 0	233	1 1 1 0 1 0 0 1

234	1 1 1 0 1 0 1 0	245	1 1 1 1 0 1 0 1
235	1 1 1 0 1 0 1 1	246	1 1 1 1 0 1 1 0
236	1 1 1 0 1 1 0 0	247	1 1 1 1 0 1 1 1
237	1 1 1 0 1 1 0 1	248	1 1 1 1 1 0 0 0
238	1 1 1 0 1 1 1 0	249	1 1 1 1 1 0 0 1
239	1 1 1 0 1 1 1 1	250	1 1 1 1 1 0 1 0
240	1 1 1 1 0 0 0 0	251	1 1 1 1 1 0 1 1
241	1 1 1 1 0 0 0 1	252	1 1 1 1 1 1 0 0
242	1 1 1 1 0 0 1 0	253	1 1 1 1 1 1 0 1
243	1 1 1 1 0 0 1 1	254	1 1 1 1 1 1 1 0
244	1 1 1 1 0 1 0 0	255	1 1 1 1 1 1 1 1



マシン語は16進数で書き表わす

16進数は、16で桁上がりする数

マシン語は、その名前の通り機械の言葉です。パソコンの内部で使われている言葉は、電気信号です。その電気信号のオンに1、オフに0を対応させた言葉が、マシン語ということになります。つまり、81頁で説明している2進数です。私たちがふつう使っているパソコンは、8ビットをひとつの単位として扱っているので、8ビットの2進数です。

ところが、マシン語でプログラムを書き表わすという場合、この2進数はいりません。16進数を使います。なぜ2進数を使わずに、16進数を使ってプログラムを書き表わすのかというと、2進数は、私たちにとって非常に理解しがたいものだからです。とって、10進数でマシン語のプログラムを書き表わしたのでは、パソコンが、それを2進数に直すのが大変です。そこで、私たちにもわかりやすく、パソコンにとっても便利な16進数を作り、16進数でマシン語のプログラムを書き表わすことになったわけです。したがって、私たちにとってのマシン語は、16進数を指しています。

さて16進数は、16を一区切りにして表わす数です。10進数が、10を一区切りにして表わす数であるのとおなじです。つまり、つぎに示すように、10進数は0から9までが1桁で、9のつぎの10で桁上がりしますが、16進数は0から15までが1桁で、15のつぎの16で桁上がりするということです。

0	1	2	3	4	5	6	7	8	9	← 10進数の1桁 10で桁上がりする						
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	→ 16進数の1桁 16で桁上がりする

16進数の9のつぎのA、B、C、D、E、Fは、10進数と、つぎのように対応しています。

A	B	C	D	E	F
↑	↑	↑	↑	↑	↑
10	11	12	13	14	15

したがって、16進数のAのつぎの10は、数字通りの10の意味ではなくて、10進数の16にあたります。

では、16進数を理解するために、2桁以降はどのように表わされるのか、もう少し16進数を取りあげて示します。カッコのなかの数字は10進数です。

10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
(16)	(17)	(18)	(19)	(20)	(21)	(22)	(23)	(24)	(25)	(26)	(27)	(28)	(29)	(30)	(31)
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
(32)	(33)	(34)	(35)	(36)	(37)	(38)	(39)	(40)	(41)	(42)	(43)	(44)	(45)	(46)	(47)
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
(48)	(49)	(50)	(51)	(52)	(53)	(54)	(55)	(56)	(57)	(58)	(59)	(60)	(61)	(62)	(63)
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
(64)	(65)	(66)	(67)	(68)	(69)	(70)	(71)	(72)	(73)	(74)	(75)	(76)	(77)	(78)	(79)
⋮															
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
(144)	(145)	(146)	(147)	(148)	(149)	(150)	(151)	(152)	(153)	(154)	(155)	(156)	(157)	(158)	(159)
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
(160)	(161)	(162)	(163)	(164)	(165)	(166)	(167)	(168)	(169)	(170)	(171)	(172)	(173)	(174)	(175)
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
(176)	(177)	(178)	(179)	(180)	(181)	(182)	(183)	(184)	(185)	(186)	(187)	(188)	(189)	(190)	(191)
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
(192)	(193)	(194)	(195)	(196)	(197)	(198)	(199)	(200)	(201)	(202)	(203)	(204)	(205)	(206)	(207)
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
(208)	(209)	(210)	(211)	(212)	(213)	(214)	(215)	(216)	(217)	(218)	(219)	(220)	(221)	(222)	(223)
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
(224)	(225)	(226)	(227)	(228)	(229)	(230)	(231)	(232)	(233)	(234)	(235)	(236)	(237)	(238)	(239)
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
(240)	(241)	(242)	(243)	(244)	(245)	(246)	(247)	(248)	(249)	(250)	(251)	(252)	(253)	(254)	(255)
100	101	102	103	104	105	106	107	108	109	10A	10B	10C	10D	10E	10F
(256)	(257)	(258)	(259)	(260)	(261)	(262)	(263)	(264)	(265)	(266)	(267)	(268)	(269)	(270)	(271)

以上16進数は、すべてFのつく1F、2F、3Fのつぎで桁上がりします。

4ビットの2進数は、16進数の1桁で表わされる

では、この16進数は、2進数とどのように対応しているでしょうか。まず、4ビットの2進数と16進数の対応表を示します。数がわかりやすいように10進数も示しておきます。

2進数	16進数	10進数
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

このように、4ビットの2進数で表わすことができる0000から1111までは、16進数1桁で表わすことができます。

4ビットの2進数=16進数1桁



8ビットの2進数は、16進数2桁で表わされる

では、8ビットの2進数を16進数で表わすとどうなるでしょうか。つぎに16進数と8ビットの2進数の対応表を示します。

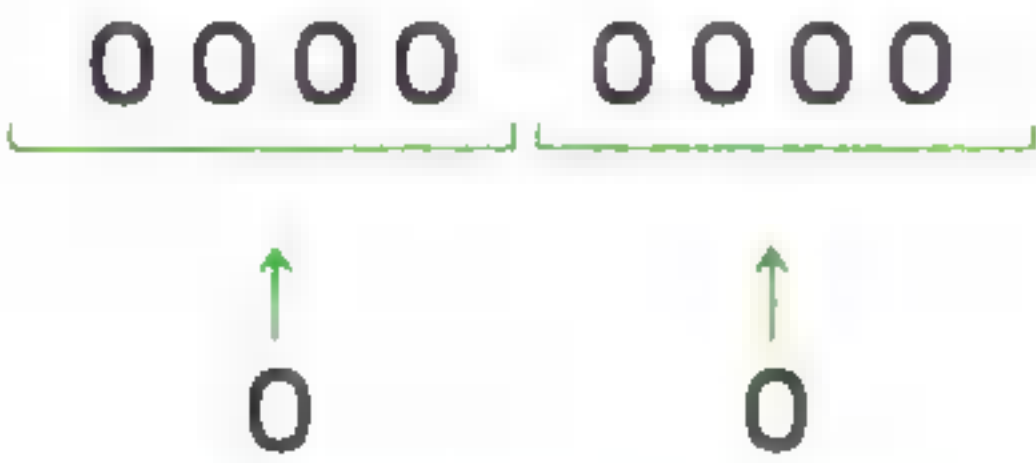
2進数	16進数	2進数	16進数
00000000	00	00001000	08
00000001	01	00001001	09
00000010	02	00001010	0A
00000011	03	00001011	0B
00000100	04	00001100	0C
00000101	05	00001101	0D
00000110	06	00001110	0E
00000111	07	00001111	0F

うえに示したように、8ビットの2進数の場合は、00、01、02といったように、16進数2桁で表わされます。

8ビットの2進数を、その2進数に対応した16進数にするには、まず、8ビットの2進数を、上位4ビット、下位4ビットの2つに区切ります。



そして、さきに示した4ビットの2進数に対応した16進数を、上位4ビット、下位4ビットの2進数にあてはめます。4ビットの2進数0000は、16進数では0ですから、上位4ビットの2進数0000に16進数の0をあてはめます。下位4ビットの2進数0000もおなじですから、16進数の0をあてはめます。



このようなわけで、8ビットの2進数00000000は、16進数00の2桁で表わされるのです。

8ビットの2進数=16進数2桁

そのほかの場合も、このようにして16進数2桁となります。2進数と16進数

の変換については、97頁で説明していますから、97頁を参照してください。

ところで、8ビットの2進数は1バイトです。したがって、16進数2桁も1バイトとなります。

8ビットの2進数=1バイト
2桁の16進数=1バイト

16進数4桁の場合は2バイトとなります。

F331 FFFF 00C3 17AB
└──────────┘ └──────────┘
16進数4桁は2バイト

おなじように、16進数6桁は3バイトとなります。

C36A00 C3576D 3F00EF
└──────────┘ └──────────┘
16進数6桁は3バイト

メモリには1バイトずつ記憶する

これまで説明したように、私たちにとってのマシン語は、16進数です。また、私たちがふつう使っているパソコンは、8ビットをひとつの単位として扱っています。したがって、マシン語を記憶するメモリも、ひとつのメモリに8ビット、つまり1バイトずつ記憶するようになっていきますから、16進数2桁ずつ記憶することになります。マシン語を入力する場合も、当然、16進数2桁ずつ入力することになります。まず、マシン語の入力の実行結果を示します。

mon
*SD000
D000 FF-21 FF-90 FF-F9 FF-3E
D004 FF-F5 FF-77 FF-76 FF-23
D008 FF-23 FF-C3 FF-05 FF-D0
D00C FF-

実行結果

←メモリには16進数2桁
ずつ入力する

4 auto on to list run

このように入力したマシン語は、つぎのようにひとつのメモリ番地に、2桁ずつ、つまり1バイトずつ記憶されます。

メモリ番地	マシン語	メモリ番地	マシン語
D000	21	D006	76
D001	90	D007	23
D002	F9	D008	23
D003	3E	D009	C3
D004	F5	D00A	05
D005	77	D00B	D0

マシン語は、このように、ひとつのメモリに16進数2桁ずつ記憶される

つぎにDコマンドで、入力したマシン語を表示させた実行結果を示します。

```

mon
*DD000
D000 21 90 F9 3E F5 77 76 23
D008 23 C3 05 D0 FF FF FF FF
*■
        
```

実行結果

ひとつのメモリには16進数2桁つまり1バイトずつ記憶する

is auto auto list run

なお、メモリ番地は、D000、D001、D002というように、16進数4桁でつけられていますから、メモリ番地は2バイトということになります。



16進数を2進数へ変換するには

2 進数・16 進数変換とビットずらし

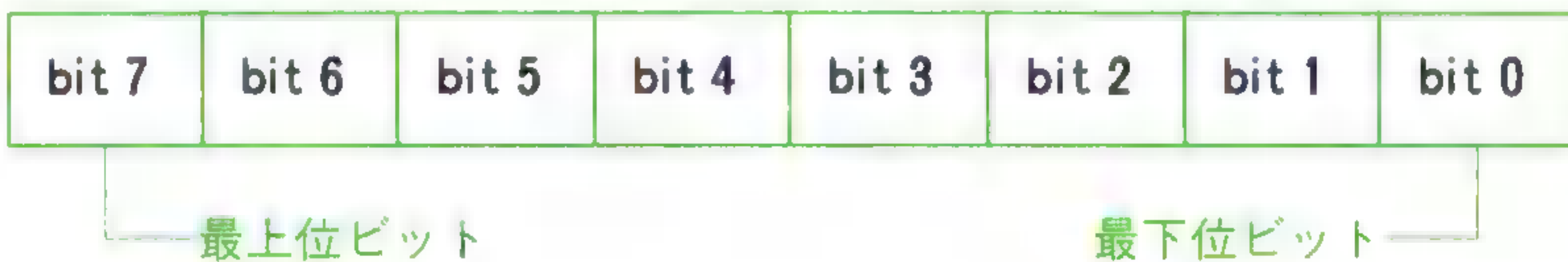
81頁で2進数について、また、91頁で16進数について説明しました。マシン語を使う場合、2進数を16進数へ、16進数を2進数へ変換する方法を知っていると、なにかと便利です。そこで、ここでは2進数を16進数へ、また16進数を2進数へ変換する方法について説明することにします。

まず、2進数を16進数へ変換することからはじめます。

2進数と16進数へ変換する

私たちが使っているパソコンは、8ビット（1バイト）をひとつの単位として扱っています。

2進数の8ビット（1バイト）を図に示すと、つぎのようになります。

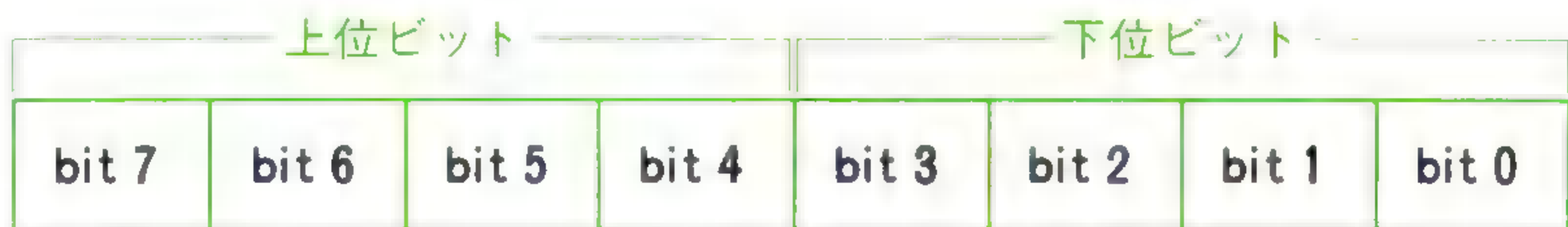


つまり、8ビットの2進数11111111（10進数の255）を例にとると、



右端の最下位ビットから最上位ビットに向かって、**bit (ビット) 0**、**bit 1**、**bit 2** と呼ばれるということです。

さらに、このbit 0 からbit 7 までは、つぎに示すように、上位ビットと下位ビットの2つに分かれます。



うえに示したように、上位ビット、下位ビットの2つに分けると、上位ビット、下位ビットともに4ビットずつになります。つまり、つぎに示すように、4ビットずつの2進数になるということです。



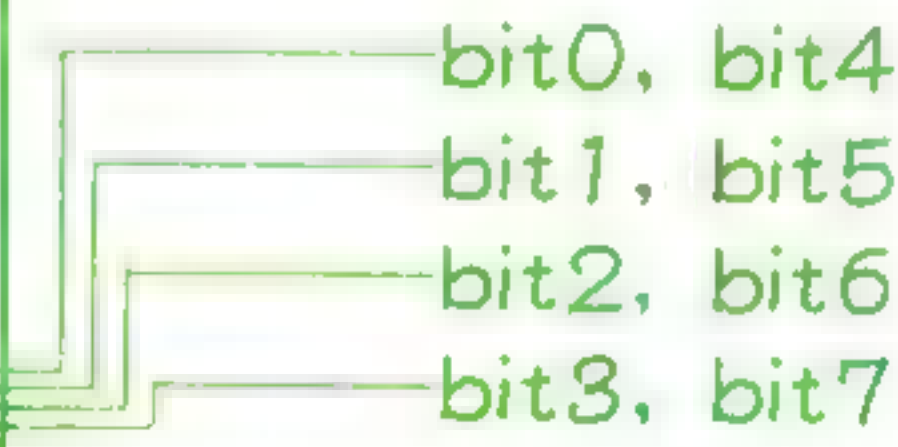
4ビットずつの2進数になる

4ビットの2進数は、1桁の16進数です。したがって、1桁の16進数を上位ビット、下位ビットにあてはめます。

この場合必要になるのが、4ビットの2進数の16進数対応表です。4ビットの2進数の16進数対応表は93頁に示してありますが、便宜上つぎに示します。

10進数	16進数	2進数
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

なぜ0011が10進数の3を表わすのかなどは81頁の2進数の表わし方で説明しています。



2進数を16進数へ、16進数を2進数へ変換するには

さて、4ビットの2進数の16進対応表のなかから、上位4ビットの1 1 1 1と、下位ビットの1 1 1 1にあてはまる1桁の16進数を捜します。

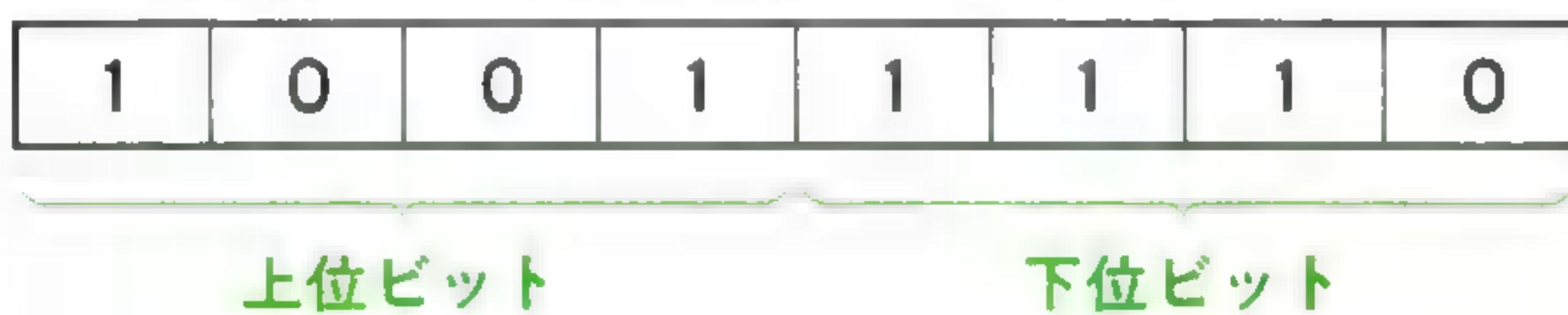
この対応表は、表の下に示しているように、4ビットの2進数の右端がbit 0とbit 4の両方になります。つぎが、bit 1とbit 5、そのつぎがbit 2とbit 6、最後がbit 3とbit 7となります。つまり、上位ビット、下位ビットの区別なく、この4ビットの2進数が、1桁の16進数に対応しているということです。

したがって、上位ビットの1 1 1 1も、下位ビットの1 1 1 1も、おなじように1桁の16進数はFとなります。そこで、この1桁の16進数Fを、つぎのように上位ビット、下位ビットにあてはめます。



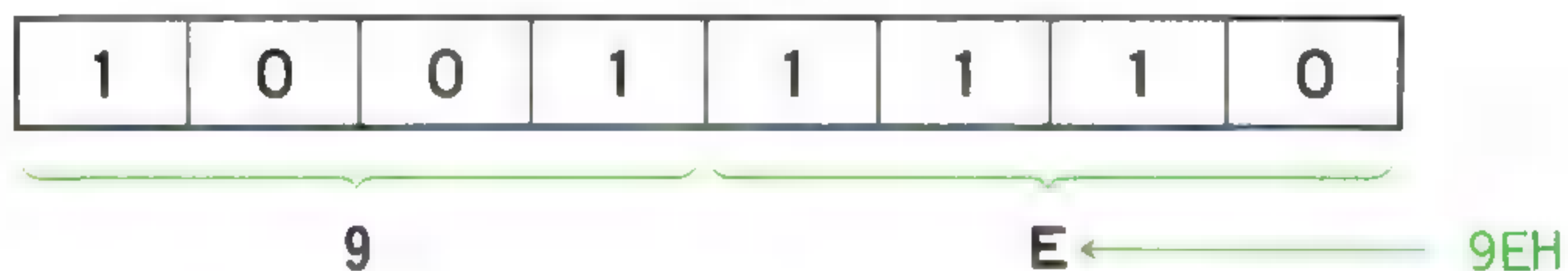
これで、8ビットの2進数11111111を16進数へ変換することができたわけです。つまり、8ビットの2進数11111111は、2桁の16進数でFFH（10進数で255）になります。FFのつぎのHは、16進数を表わす記号で、10進数と区別するために、16進数を表わすときは、Hをつけることになっています。

では、もうひとつ8ビットの2進数を、16進数に変換しておくことにします。



8ビットの2進数10011110を上位ビット、下位ビットの2つに分けると、上位ビットは1 0 0 1、下位ビットは1 1 1 0となります。上位ビットの1 0 0 1にあてはまる1桁の16進数をさきの対応表でみると9です。下位ビットの1 1 1 0はEです。

そこで、この1桁の16進数9とEを、上位ビット、下位ビットにそれぞれあてはめます。



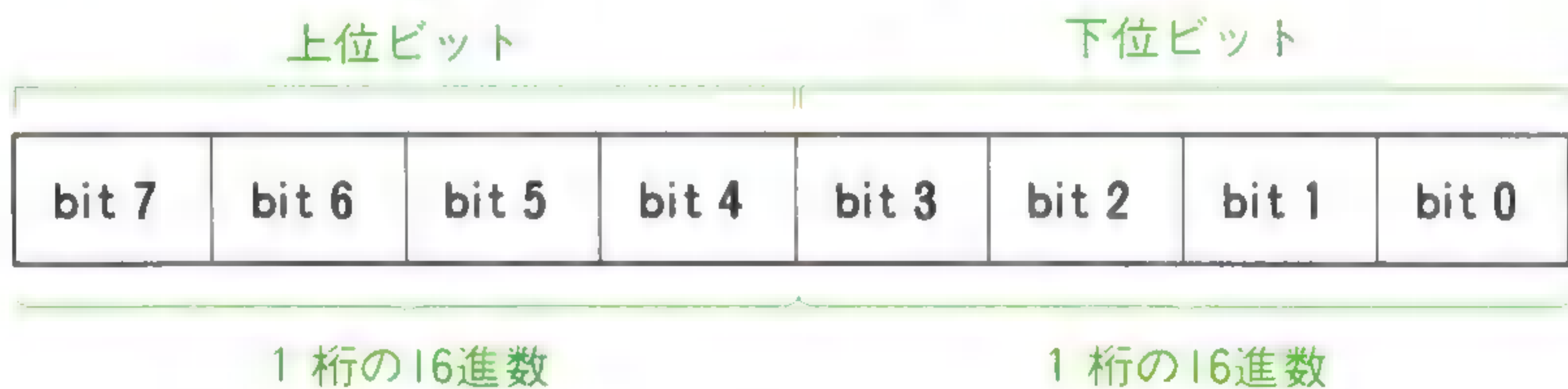
したがって、8ビットの2進数10011110は、2桁の16進数では9EH（10進数では158）ということになります。

これが、2進数を16進数に変換する方法です。

16進数を2進数に変換する

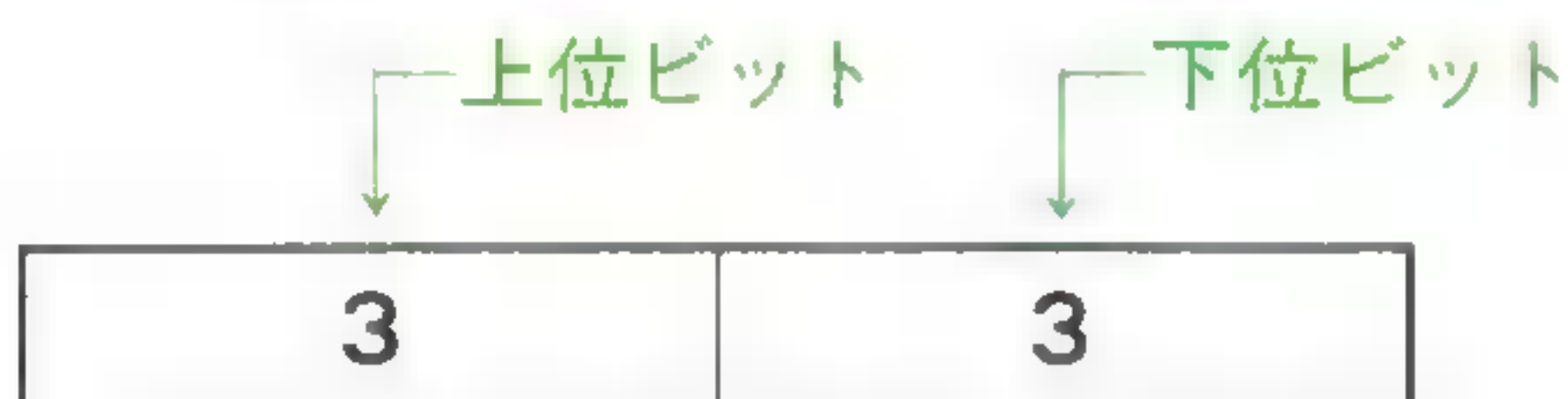
これまで、2進数を16進数へ変換する方法について説明してきました。16進数を2進数に変換するには、2進数を16進数に変換するのと、逆のことができればよいわけです。

8ビットの16進数は、つぎに示すように、上位ビットの1桁の16進数と、下位ビットの1桁の16進数の2桁で表わされます。

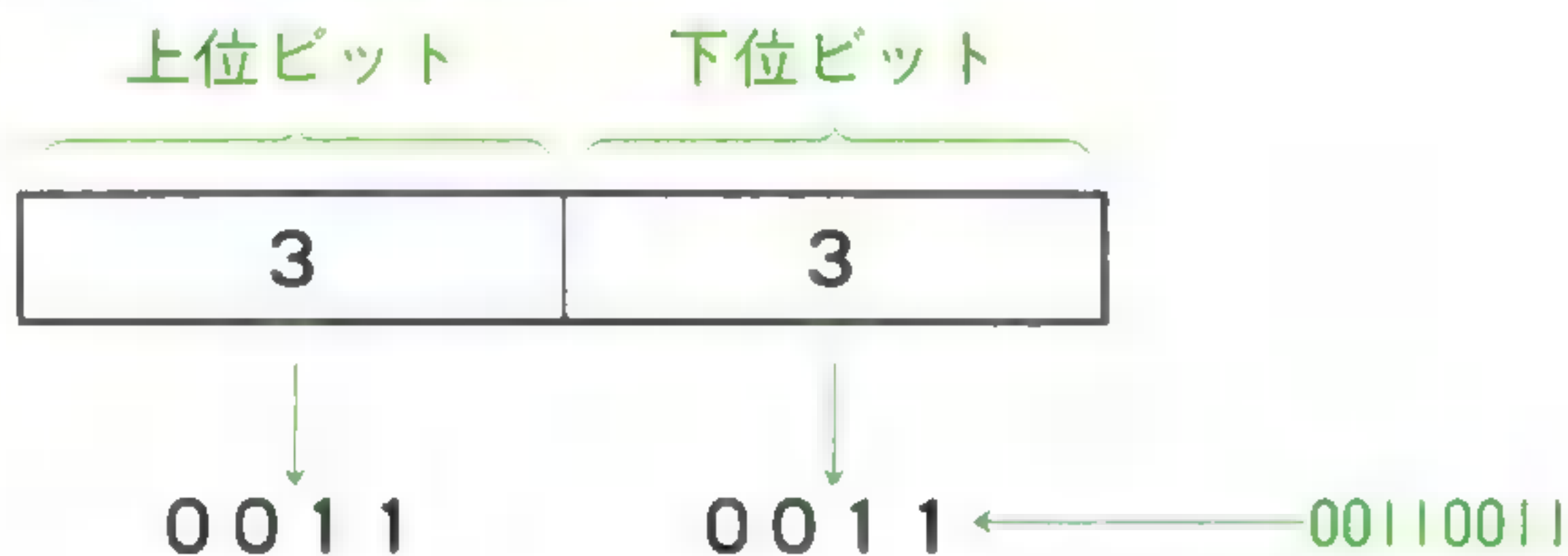


したがって、16進数を2進数に変換するには、上位ビットの1桁の16進数にあてはまる4ビットの2進数と、下位ビットの1桁の16進数にあてはまる4ビットの2進数を、さきに示した対応表から探し出して、それぞれにあてはめればよいのです。

では、16進数33（10進数で51）を2進数に変換してみることにします。



16進数33を、上位ビットと下位ビットの2つに分けると、上位ビット、下位ビットとも3Hです。そこで、3Hにあてはまる4ビットの2進数を、さきの対応表でみると0011です。この0011を、上位ビット、下位ビットの3Hにあてはめます。



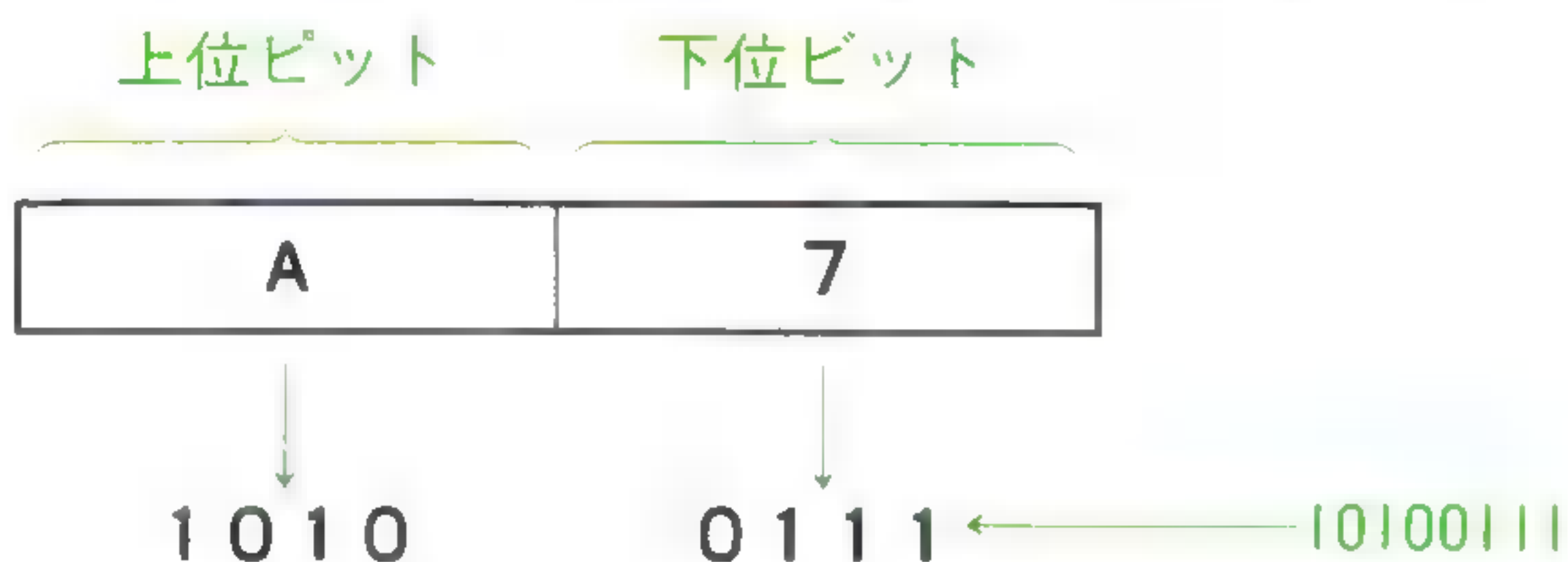
したがって16進数33は、2進数で00110011ということになります。

ではもうひとつ、16進数A7（10進数で167）を2進数に変換しておくことにします。

2進数を16進数へ、16進数を2進数へ変換するには



上位ビットAHにあてはまる4ビットの2進数をさきの対応表でみると1010です。下位ビット7Hをさきの対応表でみると0111です。そこで、1010を上位ビットに、0111を下位ビットにあてはめます。



したがって、16進数A7は、2進数で10100111ということになります。
以上が16進数を2進数に変換する方法です。



2進数を10進数へ 10進数を2進数へ変換するには

ビット番号にふられた値に基づいて変換する

97頁で2進数を16進数へ、16進数を2進数へ変換する方法について説明しました。ここでは、2進数を10進数へ変換する方法と、10進数を2進数へ変換する方法について説明します。まず、2進数を10進数に変換する方法です。

2進数を10進数に変換する

8ビットの2進数には、つぎに示すように、bit 0 からbit 7 までのビット番号がついています。そして、それぞれのビット番号に1、2、4、8、16、32、64、128の値をふります。

128	64	32	16	8	4	2	1
↓	↓	↓	↓	↓	↓	↓	↓
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

ビット番号に置かれている値は、84 頁で説明している8ビットの2進数の表わし方で、つぎのようにメモリを刻んで、そのメモリにふった値とおなじです。この場合は、メモリをビット番号に変更しているだけです。

128	64	32	16	8	4	2	1
+---	+---	+---	+---	+---	+---	+---	+

2進数を10進数に変換する場合は、このビット番号にふられた値に基づいて行います。では、2進数11111111を10進数に変換することにします。2進数11111111をビット番号にあてはめると、つぎのようになります。

1	1	1	1	1	1	1	1
↓	↓	↓	↓	↓	↓	↓	↓
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
↑	↑	↑	↑	↑	↑	↑	↑
128	64	32	16	8	4	2	1

2進数を10進数へ、10進数を2進数へ変換するには

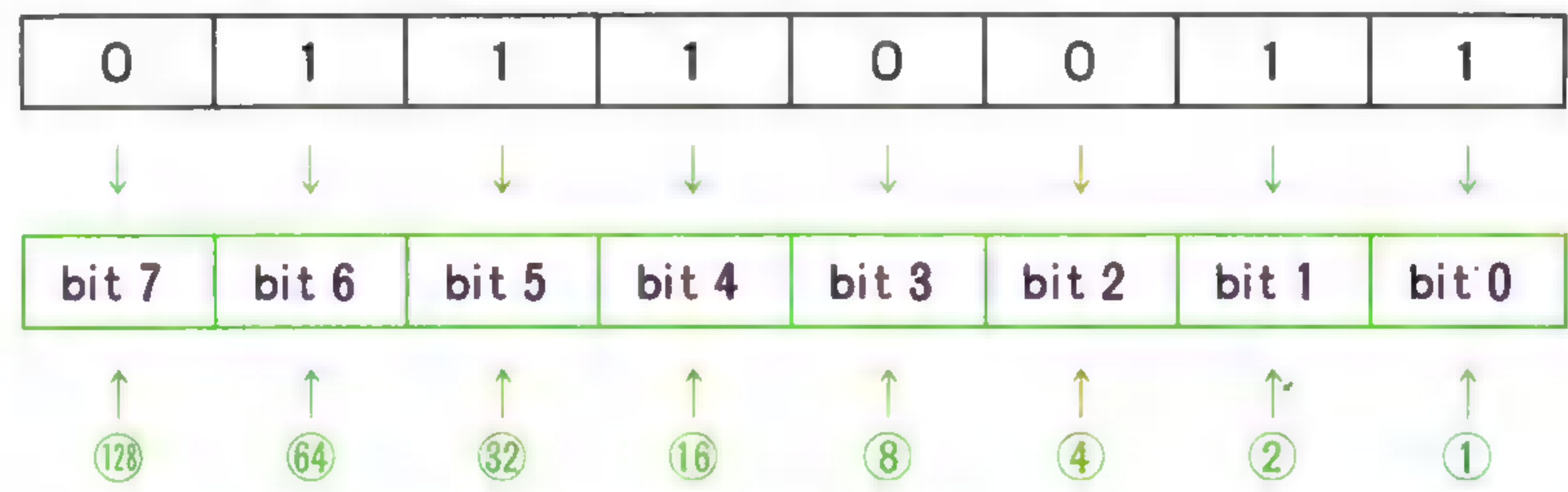
2進数11111111の場合は、すべてのビット番号に1が置かれていますから、それぞれのビット番号にふられている値を足します。

$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$

したがって2進数11111111は、10進数で255ということになります。

この場合は、11111111なので、全部のビット番号に1が置かれましたが、ビット番号に1ではなくて0が置かれた場合は、0が置かれたビット番号の値を0にして足します。つぎに、その例を示します。

では、2進数01110011を10進数に変換してみることにします。2進数01110011を、ビット番号にあてはめると、つぎのようになります。

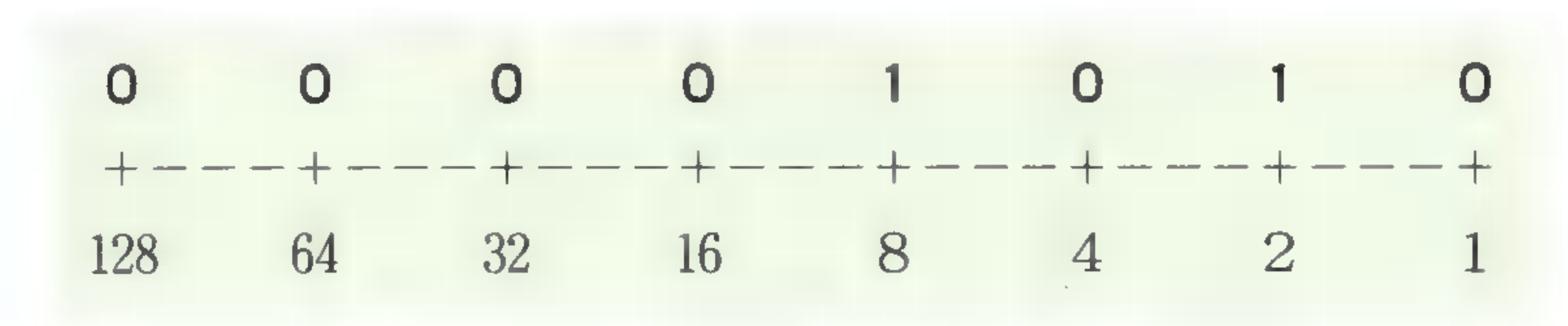


この場合、0が置かれているビット番号の値は0にしますから、ビット番号2が0、ビット番号3が0、ビット番号7が0になります。したがってつぎのようにして足すことになります。

$0 + 64 + 32 + 16 + 0 + 0 + 2 + 1 = 115$

もちろん、64 + 32 + 16 + 2 + 1 でもかまいません。

つぎに、ビット番号をいちいち使っていたのではめんどうなので、さきにしたメモリを使って、2進数00001010を10進数に変換してみることにします。メモリにふられている値は、さきの場合とおなじです。まず、2進数00001010を、つぎのようにメモリに置きます。



そして、1が置かれているメモリの値を足します。

$8 + 2 = 10$

2進数00001010は、10進数の10ということになります。

以上が、2進数を10進数に変換する方法です。

2進数を10進数に変換する方法は、ほかにむずかしい方法があります。たとえば、つぎのように計算して、2進数を10進数に変換する方法です。

$$1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

このようなむずかしい方法を、なにも好んで使うことはないので、この方法については説明を省略します。

10進数を2進数に変換する

さてつぎに、10進数を2進数に変換する方法です。

10進数を2進数に変換するには、2進数を10進数に変換したことの逆のこ
とを行えばいいのです。ここでは、ビット番号の代わりにメモリを使います。
メモリにふる値は2進数を10進数に変換したときとおなじ値です。

+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+
128				64				32				16				8				4			2	1

では、このメモリを使って、どのように10進数を2進数に変換するのか、具
体的に10進数を2進数に変換して説明することにします。まず、10進数27を2
進数に変換します。メモリには、27という値はふられていませんから、27を越
えないで最も大きい値を、メモリにふられている値のなかから選びます。する
と、16ということになりますから、16の値のメモリに1を置きます。

				1																				
+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+
128				64				32				16				8				4			2	1

27から16を引くと残りは11です。11という値はメモリにふられていませんか
ら、11を越えないで最も大きい値を、メモリにふられている値のなかから選
びます。すると8ということになりますから、8の値のメモリに1を置きます。

				1				1																
+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+
128				64				32				16				8				4			2	1

11から8を引いた残りは3です。2と1を足せば3になりますから、2と1
の値のメモリに1を置きます。

				1				1								1				1				
+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+
128				64				32				16				8				4			2	1

2進数を10進数へ、10進数を2進数へ変換するには

1 が置かれていない値のメモリのところには、0 が置かれます。したがって、つぎのようになります。

0	0	0	1	1	0	1	1
+---	+---	+---	+---	+---	+---	+---	+
128	64	32	16	8	4	2	1

つまり10進数の27を2進数に変換すると、00011011ということになります。

10進数を2進数に変換する方法は、もう理解できたことと思いますが、もう幾つかの例で10進数を2進数に変換してみます。

ではつぎに、10進数125を2進数に変換します。メモリには125はふられていませんから、メモリにふられている値のなかから、125を越えないで最も大きい値を選びます。すると64ということになりますから、64の値のメモリに1を置きます。

	1						
+---	+---	+---	+---	+---	+---	+---	+
128	64	32	16	8	4	2	1

125から64を引くと、残りは61です。61という値はメモリにふられていませんから、61を越えないで最も大きい値を選びます。すると32ということになりますから、32の値のメモリに1を置きます。

	1	1					
+---	+---	+---	+---	+---	+---	+---	+
128	64	32	16	8	4	2	1

61から32を引くと、残りは29です。29という値はメモリにふられていませんから、29を越えないで最も大きい値を選びます。すると16となりますから、16の値のメモリに1を置きます。

	1	1	1				
+---	+---	+---	+---	+---	+---	+---	+
128	64	32	16	8	4	2	1

29から16を引くと、残りは13です。ここまでくれば、あとは簡単です。メモリにふられている値のなかから、合計して13になる値を選び出して、その値のメモリに1を置けばよいわけです。8と4と1を足すと13になりますから、8と4と1の値のメモリに1を置きます。1が置かれないメモリのところは0を

置きます。したがって、つぎのようになります。

0	1	1	1	1	1	0	1
+---	+---	+---	+---	+---	+---	+---	+
128	64	32	16	8	4	2	1

つまり10進数125を2進数に変換すると、01111101ということになります。

では最後に、10進数240を2進数に変換することにします。240という値はメモリにふられていませんから、最も大きい128に1を置きます。

1							
+---	+---	+---	+---	+---	+---	+---	+
128	64	32	16	8	4	2	1

240から128を引くと、残りは112です。112という値はメモリにふられていませんから、つぎに64の値のメモリを選んで1を置きます。

1	1						
+---	+---	+---	+---	+---	+---	+---	+
128	64	32	16	8	4	2	1

112から64を引くと、残りは48です。48という値はメモリにふられていませんから、つぎは32の値のメモリを選んで1を置きます。

1	1	1					
+---	+---	+---	+---	+---	+---	+---	+
128	64	32	16	8	4	2	1

48から32を引くと、残りは16です。16という値はメモリにふられていますから、16の値のメモリに1を置きます。1が置かれないメモリには0を置きます。

1	1	1	1	0	0	0	0
+---	+---	+---	+---	+---	+---	+---	+
128	64	32	16	8	4	2	1

つまり10進数240を2進数に変換すると、11110000ということになります。

この10進数を2進数に変換する方法は、慣れていないと、ちょっとまごつくかもしれませんが、慣れれば非常に簡単な方法です。10進数を2進数に変換する方法で、これ以上簡単で、わかりやすい方法はありませんから、使って慣れることです。

マシン語のプログラムを書くには

アセンブリ言語、アセンブル、アセンブラ

私たちにとってのマシン語は、2進数ではなくて16進数です。したがってマシン語のプログラムは、つぎのように16進数で書かれたプログラムをいいます。

3E E9 32 E4 F6 76

このマシン語のプログラムは、わずか6バイトですが、これでも、あることを行うマシン語のプログラムです。どんなことをさせるマシン語のプログラムか、ベーシックのプログラムで示します。

```
10 LOCATE 18.8  
20 PRINT "♥"  
30 END
```

ベーシックのプログラムを見ればわかるように、画面の列番号18、行番号8の位置に、ハートを表示するプログラムです。

ところで、このマシン語のプログラムが6バイトでできているというのは、16進数2桁が1バイト（8ビット）で、それが6個あるからです。また、16進数2桁ずつで、マシン語のプログラムを書いていくのは、パソコンが1バイト、つまり8ビットをひとつの単位として扱っているからです。

まず、アセンブラでプログラムを書く

さて、うえに示したようなマシン語のプログラムを書く場合、直接、16進数で書いていくわけではありません。うえに示した、わずか6バイトのマシン語のプログラムを見ただけでもわかるように、直接、16進数だけを並べて書くなどということは、とてもできるものではないからです。

では、どのようにしてマシン語のプログラムを書くのでしょうか。

まず、アセンブリ言語というものを使ってプログラムを書きます。

アセンブリ言語は、マシン語を私たちにわかりやすくした言語で、マシン語のひとつひとつの命令の意味を、英語の略語で表わしたものです。

このアセンブリ言語でプログラムを書いてから、それを、16進数を使ってマシン語のプログラムに書きなおします。

ではつぎに、前に示したマシン語のプログラムのもとになる、アセンブリ言語で書いたプログラムを示します。

```
LD  A, E9H
```

```
LD  (F6E4H), A
```

```
HALT
```

これが、アセンブリ言語で書いたプログラムです。

アセンブルは、アセンブリ言語をマシン語になおすこと

さて、このアセンブリ言語で書いたプログラムを、マシン語のプログラムになおすわけですが、マシン語のプログラムに書きなおすことを**アセンブル**、**アセンブラ**といいます。**アセンブル**とは、アセンブリ言語で書いたプログラムを、手作業でマシン語のプログラムに書きなおすことをいいます。**アセンブラ**とは、アセンブリ言語で書いたプログラムを、マシン語に翻訳するプログラムを使って、マシン語のプログラムに書きなおすことをいいます。

私たちがこの本で、アセンブリ言語で書いたプログラムを、マシン語のプログラムに書きなおすのは、アセンブルです。つまり、手作業でマシン語になおしていくということです。

アセンブルには、Z-80アセンブル表が必要

アセンブリ言語をマシン語になおす、つまりアセンブルを行うのに必要なものがあります。それは**Z-80アセンブル表**です。**Z-80アセンブル表**は、アセンブリ言語で書いたプログラムを、どのようなマシン語にアセンブルするかを指示したものです。アセンブルを行うときは、この**Z-80アセンブル表**にしたがって行います。**Z-80アセンブル表**は、この本の巻末に掲載してあります。

さきに示したアセンブリ言語のプログラムの説明、アセンブルの仕方、また**Z-80アセンブル表**の活用の仕方については、つぎの頁から、ひとつひとつ説明していきます。したがって、ここではマシン語のプログラムを書く場合に必要とする事柄をおぼえておいてください。

画面への表示

LDは、ベーシックのLETと似た働き

画面にキャラクタを表示する場合、ベーシックでは、つぎに示すように、PRINT文だけですみます。

```
10 PRINT "♥"  
20 END
```

マシン語の場合は、このように簡単にはいきません。

```
LD  A, E9H  
LD  (F300H), A  
HALT
```

これが、ハートを画面に表示するプログラムです。このプログラムはアセンブリ言語で書かれています。このように、アセンブリ言語でプログラムを書いたから、つぎにZ-80アセンブル表を使ってアセンブルするわけですが、そのまえに、アセンブリ言語の命令を知っていなければなりません。

LDは、ロード命令という

うえに示したアセンブリ言語のプログラムの1行目と、2行目の最初にLDとあります。LDは命令で、ロード命令といいます。

LD命令は、つぎのような形をしています。

```
LD  X1, X2
```

LD命令は、ベーシックの命令でいうと、LET命令とすることができます。ベーシックのLET命令は、現在では使わなくてもよいので、LET命令を知らない人もいないかもしれませんが、LET命令は、つぎに示すように、

```
LET  X1=X2
```

=の右側にあるX2を、=の左側のX1に代入する働きをします。

したがって、LD命令も、つぎに示すように、

LD X1 ← X2

X2をX1に代入する働きをします。

Aは、Aレジスタ

さて、LD命令は、X1にX2を代入する命令であることがわかりました。
したがって、1行目の場合は、

LD A, E9H

ですから、AへE9Hを代入するということになります。

ではつぎは、このAです。マシン語では変数を使いません。変数でないとすると、Aはなんでしょう。Aは、CPUにあるAレジスタのことです。

CPUには、Aレジスタのほかに、B、C、D、E、H、Lのレジスタがあります。CPUにあるこれらのレジスタは、CPUに取り込んだ命令やデータなどの情報を、一時記憶しておく場所です。

レジスタ=CPUに取り込んだ情報を一時記憶しておく場所

また、レジスタは、記憶しておくことできる情報量が決まっています。Aレジスタは、8ビット・レジスタです。したがって、8ビットの情報を、一時記憶します。

Aレジスタ=8ビット・レジスタ

このように、Aレジスタは8ビット・レジスタと決まっていますから、

LD A, 12H

とすることはできても、つぎのようにはできません。

LD A, 1234H

なぜ、このようにすることができないかわかると思いますが、1234Hは2バイト、つまり、16ビットだからです。

8ビット（1バイト）=16進数2桁

ところで、Aレジスタ以外のB、C、D、E、H、Lレジスタは、ひとつひとつの場合は8ビット・レジスタですが、B、C、D、E、H、Lレジスタは2つ組み合わせることができますから、2つ組み合せて、16ビット・レジスタとして使うことができます。

このことは、あとで説明することにします。

E9Hは、ハートのキャラクタコード

では、E9Hです。E9Hは、ハートのキャラクタコードです。

E9H=ハートのキャラクタコード

キャラクタコード表をみると、ハートの上位ビットはE、下位ビットは9、となっています。つまり、E9です。16進数は、10進数と区別するため、16進数のあとには、Hをつけることになっていますから、E9Hとなるわけです。これまで説明してきたことから、1行目の意味は、

LD A, E9H

AレジスタにハートのキャラクタコードE9Hを代入する
ということになります。

LD (F300H), Aとは

では、2行目です。

LD (F300H), A

LD命令は、LD X1, X2という形で、X2をX1に代入するという命令です。この場合、X1が(F300H)で、X2がAですから、Aレジスタに記憶されているものを(F300H)に代入することになります。

この場合は、Aレジスタには、1行目でハートのキャラクタコードE9Hを代入しました。したがって、Aレジスタに記憶されているハートのキャラクタコードE9Hを(F300H)に代入することになります。

さて、(F300H)とはなんでしょう。カッコのなかのF300Hは、Hがついているので、16進数であるということはわかります。また、16進数は2桁が1バイト(8ビット)ですから、

(F300H)

は、2バイトの16進数です。

LD命令のすぐあとに、

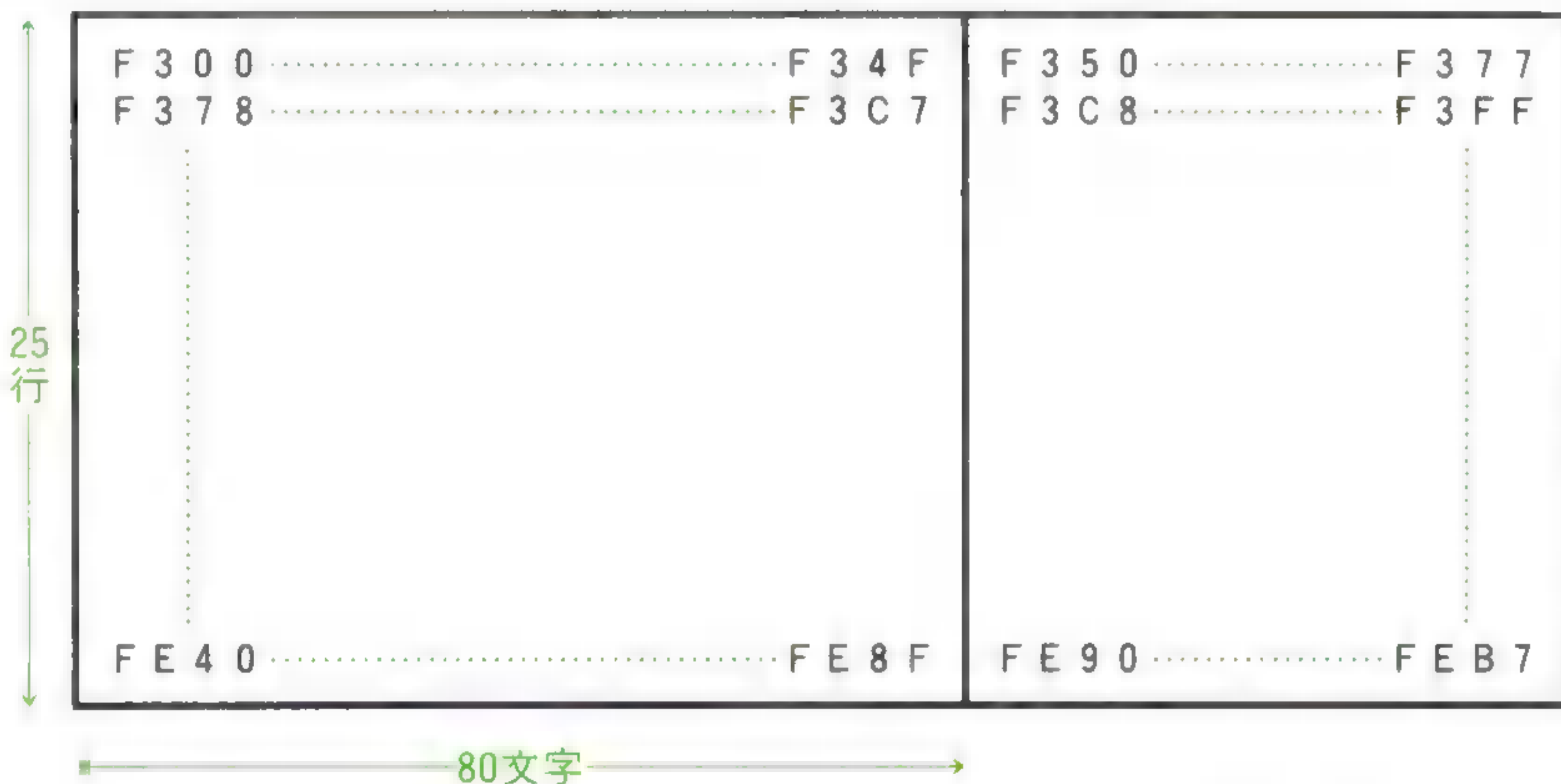
LD (F300H), A

と、カッコで囲まれた2バイト(4桁)の16進数が置かれたときは、その2バイト(4桁)の16進数は、メモリ番地を示しています。

2バイトの16進数がメモリ番地を示しているとすると、つぎは、F300番地がどこにあるかが問題となります。

つぎの図に示すように、画面の一番左端上です。

VRAMの構造(PC-800I)



Aレジスタに記憶されているものを代入すると、ハートが指定されたVRAMのメモリ番地に表示されることになります。

これまで、説明してきたことから、2行目の意味は、

LD (F300H), A

Aレジスタに記憶されているハートを、F300番地に表示するということになります。

VRAMのひとつひとつのメモリ番地を、表にして巻末に示しました。

HALT 2. END

HALTは、154頁で説明しているように、ベーシックのENDとおなじです。ただし、ベーシックの場合はENDを省略することができますが、マシン語の場合は省略することができません。HALTを省略するとエラーになりますから注意してください。

パソコンがHALTを実行すると、HALTを実行したところで、CPUの働きが停止します。

HALTを実行して、CPUが停止したときは、どんなキーを叩いても無駄です。その場合は、リセットボタンを押します。リセットボタンを押すと、パソコンは電源を入れた最初の状態に戻ります。

このリセットボタンを押すと、メモリに記憶されているベーシックのプログラムは消滅してしまいますが、マシン語はメモリに記憶されたまま残っています。したがってふたたびマシン語のプログラムを実行させることができます。

アセンブリ言語をマシン語になおす

アセンブリ言語の意味が理解できたところで、アセンブリ言語で書いたプログラムをマシン語になおします。つまり、アセンブルです。アセンブルは、Z-80アセンブル表にもとづいて行っていきます。Z-80アセンブル表は、巻末に示してあります。したがって、巻末のZ-80アセンブル表を見ながら、アセンブルすることになります。では、1行目のアセンブルです(246頁参照)。

LD A, E9H

この場合は、アセンブル表の8ビットのところを見ます。1行目の欄は×となっていて、その右側にいろいろな項目が示されています。そのつぎの欄は、LD A, ×となっています。LD A, E9というように、LDのつぎがAの場合は、LD A, ×の欄を使います。LD A, ×の×には、1行目の欄に並んでいる項目のなかからあてはまるものを選び出します。

この場合は、LD AのあとがE9というように1バイトの数ですから、nの項ということになります。nは、1バイトの数であることを示しています。

そこで、LD A, ×とnの変わったところをみます。

LD A, ×とnの変わったところには、

3E n

とあります。このnには、1バイトの数E9Hをそのままあてはめますから、

LD A, E9H



3E E9

というように、アセンブルされます。つぎは、2行目のアセンブルです。

LD (F300H), A

この場合は、LDのあとにカッコで囲まれたメモリ番地、つまり2バイトの数がきますから、8ビットのZ-80アセンブル表のLD(nn), ×の欄を使います。nは1バイトの数を示し、nnは2バイトの数を示しています。

LD (nn), ×の×は、この場合はAですから、LD (nn), ×とAが変わったところをみます。そこには、

32 nn

とあります。このカッコのなかの **nn** には、2 バイトの数 **F300H** をそのままあてはめますから、

LD (F300H), A

32 00 F3

というように、アセンブルされます。

なにか変だと思わなかったでしょうか。カッコのなかの **F300H** をアセンブルしたとき、**00 F3** としました。

F300

00 F3

パソコンに使われている CPU が **Z-80CPU** の場合、数が 2 バイトのときは、下位 1 バイト（下 2 桁）がさきに、上位 1 バイト（上 2 桁）があとと決まっています。したがって、つぎのように、

F300

F3 00

とアセンブルすると、**F300** という数とは異なる **00F3** という数になってしまいます。この場合は、**00F3** というメモリ番地はありませんから暴走することになります。したがって、2 バイトの数をアセンブルするときは、必ず、下位 1 バイトをさきに、上位 1 バイトをあとにするようにしなければなりません。最後は、3 行目の **HALT** です。**HALT** は、CPU コントロール（247 頁参照）のところを見ます。

HALT

76

ですから、**HALT** は、**76** とアセンブルされます。

つぎに、アセンブルしたプログラムをまとめて示します。

3E E9
32 00 F3
76

メモリ上に割りつける

アセンブルが終わったら、マシン語をメモリ上に割りつけます。つまり、何番地のメモリから入力するか決めるわけです。この場合は、D000番地から割りつけることにします。

したがって、右の表のようになります。

メモリ番地	マシン語
D000	3E
D001	E9
D002	32
D003	00
D004	F3
D005	76

つぎに、このマシン語をD000番地から入力して、実行した結果を示します。

■ F300番地にハートを表示する

実行結果

mon

*GD000 ← 実行命令

h..... auto..... auto list..... func.....

つぎに、このマシン語のダンプリストを示します。

D000 3E E9 32 00 F3 76 FF FF

あるメモリ番地からあるメモリ番地へジャンプさせるには

JPは、ベーシックのGOTO

109頁で、つぎに示すマシン語のプログラムで、画面にハートを表示しました。

アセンブリ言語	マシン語
LD A, E9H	3E E9
LD (F300H), A	32 00 F3
HALT	76

このマシン語のプログラムでは、最後がHALTで終わっているので、プログラムを実行させたあと、いちいちパソコンのリセットボタンを押して、ベーシックのコマンド・レベルに戻しました。これでは、ふたたびマシン語のコマンド・レベルにすると、MON命令を入力しなければならないので非常にめんどうです。そこで、プログラムを実行させたあと、ふたたびマシン語のコマンドが受けつけられるようにします。

そうするには、マシン語のコマンドを受けつけるマシン語モニタに戻さなければなりません。マシン語のコマンドを受けつけるマシン語モニタは、5C66番地に記憶されているので、プログラムの最後のメモリ番地から、5C66番地にジャンプさせます。つまりベーシックで、ある行番号のステートメントから、ある行番号のステートメントにジャンプさせるのとおなじです。

この場合、ベーシックでは、GOTO命令を使いますが、マシン語ではJP命令を使います。

JP命令は、無条件ジャンプ

JP命令は、ジャンプ命令といいます。JP命令は、ベーシックのGOTO命令とおなじ働きですから、無条件ジャンプです。

JP命令の形は、つぎのような形をしています。

JP nn

あるメモリ番地からあるメモリ番地へジャンプさせるには

JP命令のあとのnnには、ジャンプさきのメモリ番地を書きます。nは1バイト、nnは2バイトですから、4桁の16進数を示しています。なぜ2バイトの16進数なのかわかっていると思いますが、メモリ番地は、すべて2バイトだからです。

このように、JP命令のあとに2バイトの16進数、つまりメモリ番地を書くと、そのメモリ番地に無条件にジャンプします。

JP命令で、マシン語モニタに戻す

では、さきに説明したように、ふたたびマシン語のコマンドが受けつけられるように、JP命令を使ってマシン語モニタにジャンプさせることにします。

ここでは、画面の真中にダイヤを表示させることにします。画面の真中にダイヤを表示させるプログラムをベーシックで書くと、つぎのようになります。

```
10 LOCATE 18,8
20 PRINT "◆"
30 END
```

では、これをアセンブリ言語で書きます。

109頁でハートを画面に表示しました。ここではハートではなくてダイヤですから、ハートをダイヤに変えます。したがってダイヤを画面に表示するには、まずLD命令を使って、AレジスタにダイヤのキャラクタコードEAHを代入します。キャラクタコードの調べ方は、254頁を参照してください。

LD A, EAH

つぎは、LD命令でAレジスタに記憶されているダイヤを、カッコのなかにVRAMのメモリ番地を指定して代入します。

この場合は画面の真中ですから、250頁に示した横40文字のときのVRAMのメモリ番地表をみて、真中と思われるメモリ番地を選び出して、カッコのなかに書けばいいのですが、ベーシックのプログラムでは、列番号18、行番号8としましたから、それに合せてみることにします。この位置は、VRAMのメモリ番地表ではF6E4番地あたりですから、このF6E4番地を書きます。

LD (F6E4H), A

109頁のマシン語のプログラムでは、このあとHALT(END)でした。HALTに変えて、JP命令でマシン語モニタにジャンプさせます。マシン語モニタは

5 C 6 6番地に記憶されています。したがって、JP命令のあとに5 C 6 6番地を書けばよいわけです。

JP 5 C 6 6 H

これでアセンブリ言語のプログラムができました。つぎに、これをアセンブルするわけですが、そのまえに、アセンブリ言語のプログラムを、まとめて示します。

LD A, EAH ← AレジスタにダイヤのキャラクタコードEAHを代入

LD (F6E4H), A ← Aレジスタの内容を、F6E4番地に代入

JP 5 C 6 6 H ← 5C66番地にジャンプ

アセンブリ言語をマシン語に直す

さて、アセンブルです。1行目と2行目は、109頁で説明したこととおなじです。つまり、1行目は、

LD A, EAH

です。LDのあとにAがきたときは、アセンブル表のLD A, ×の欄を使います。またこの場合は、LD A, のあとにEAHという1バイトの16進数がありますから、LD A, ×とnが変わったところをみます。

3 E n

nのところには、ダイヤのキャラクタコードを、そのままあてはめます。したがって、1行目は、

LD A, EAH



3 E EA

とアセンブルされます。2行目は、

LD (F6E4H), A

というように、LDのあとにカッコで括られた2バイトの16進数（メモリ番地）がきますから、LD (nn), ×の欄を使います。そして、LD (nn), のあとにはAがありますから、LD (nn), ×とAの変わったところをみます。

3 2 n n

nnには、カッコのなかの2バイトの16進数をそのままあてはめますから、2行目は、

あるメモリ番地からあるメモリ番地へジャンプさせるには

LD (F6E4H), A



32 E4 F6

というように、アセンブルされます。2 バイトの場合は、下位バイトの4EHがさきに、上位バイトのF6Hがあとになります。さて、3行目です。

JP 5C66H

JPのマシン語はC3です。C3のあとには、2 バイトのメモリ番地を書きますから、

C3 nn

となります。この場合、nnにはマシン語モニタのメモリ番地5C66Hを書きます。したがって、3行目は、

JP 5C66H



C3 66 5C

といったように、アセンブルされます。5C66Hは、下位バイト66Hがさきに、上位バイト5CHがあとになります。

つぎに、アセンブルしたプログラムをまとめて示します。

3E	EA	←	LD A, EAH	
32	E4	F6	←	LD (F6E4H), A
C3	66	5C	←	JP 5C66H

メモリ上に割りつける

アセンブルが終わったら、マシン語をメモリ上に割りつけます。つまり、何番地のメモリ番地から入力するか決めるわけです。メモリ番地D000から入力する場合はD000番地から、マシン語をひとつひとつ置きます。また、C000番地から入力する場合はC000番地から、マシン語をひとつひとつ置きます。メモリ上の割りつけ方については、115頁を参照してください。

つぎに、D000番地から入力したマシン語のダンプリストを示します。

D000	3E	EA	32	E4	F6	C3	66	5C
メモリ番地	マシン語							

つぎに示すのが、実行結果です。***GD000**のつぎに1行空白があって、そのつぎに***印**が表示されています。この***印**が、ダイヤを画面に表示したあと、JP命令で**5C66**番地のマシン語のモニタにジャンプして表示した***印**で、ふたたびマシン語のコマンドを受けつけるための***印**です。

mon
*GD000

実行結果

*■

JP命令で5C66番地にジャンプして表示する

◆

LD (F6E4H), Aで表示する

h

auto

so to list

run

ベーシックのコマンド・レベルに戻すには

マシン語のプログラムを実行したあと、ベーシックのコマンド・レベルに戻すには、つぎのように、さきのJP 5C66Hを、JP 0000Hに変更します。つまり、JP命令で0000番地にジャンプさせます。

JP 5C66H C3 66 5C

■ ■

JP 0000H C3 00 00

つぎに示すのが、JP 0000H (C3 00 00) に変更したダンプリストです。

D000 3E EA 32 E4 F6 C3 00 00

このようにすると、プログラムを実行してダイヤを表示したあと、つぎの実行結果に示すように、ベーシックのコマンド・レベルに戻ります。

NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

実行結果

Ok



JP命令で0000番地にジャンプ
させると、電源を入れた状態に戻る

auto auto list run

ただし、ベーシックのプログラムがマシン語のプログラムと一緒に入っている場合、JP命令で0000番地にジャンプさせると、ベーシックのプログラムは消滅してしまいます。したがって場合によっては、この方法では困ることがあります。マシン語のプログラム同様、ベーシックのプログラムも消滅させないで、ベーシックのコマンド・レベルに戻すには、つぎに説明する方法を使います。

ベーシックのプログラムをも消滅させない方法

さきのように、JP命令で0000番地にジャンプさせると、ベーシックのコマンド・レベルには戻りますが、ベーシックのプログラムは消滅してしまいます。マシン語のプログラムとともに、ベーシックのプログラムも消滅させないで、ベーシックのコマンド・レベルに戻すには、JP命令で0008番地にジャンプさせます。JP 0008Hをアセンブルすると、つぎのようになります。

JP 0008H



C3 08 00

では、実際にプログラムを実行させて、その様子を見てみることにします。つぎに示すベーシックとマシン語のプログラムはなんの関連もありますが、このふたつを入力して、マシン語のプログラムを実行してみます。


```
10 PRINT "JP テ" 0008 ハンチウ ジャンプ"
```

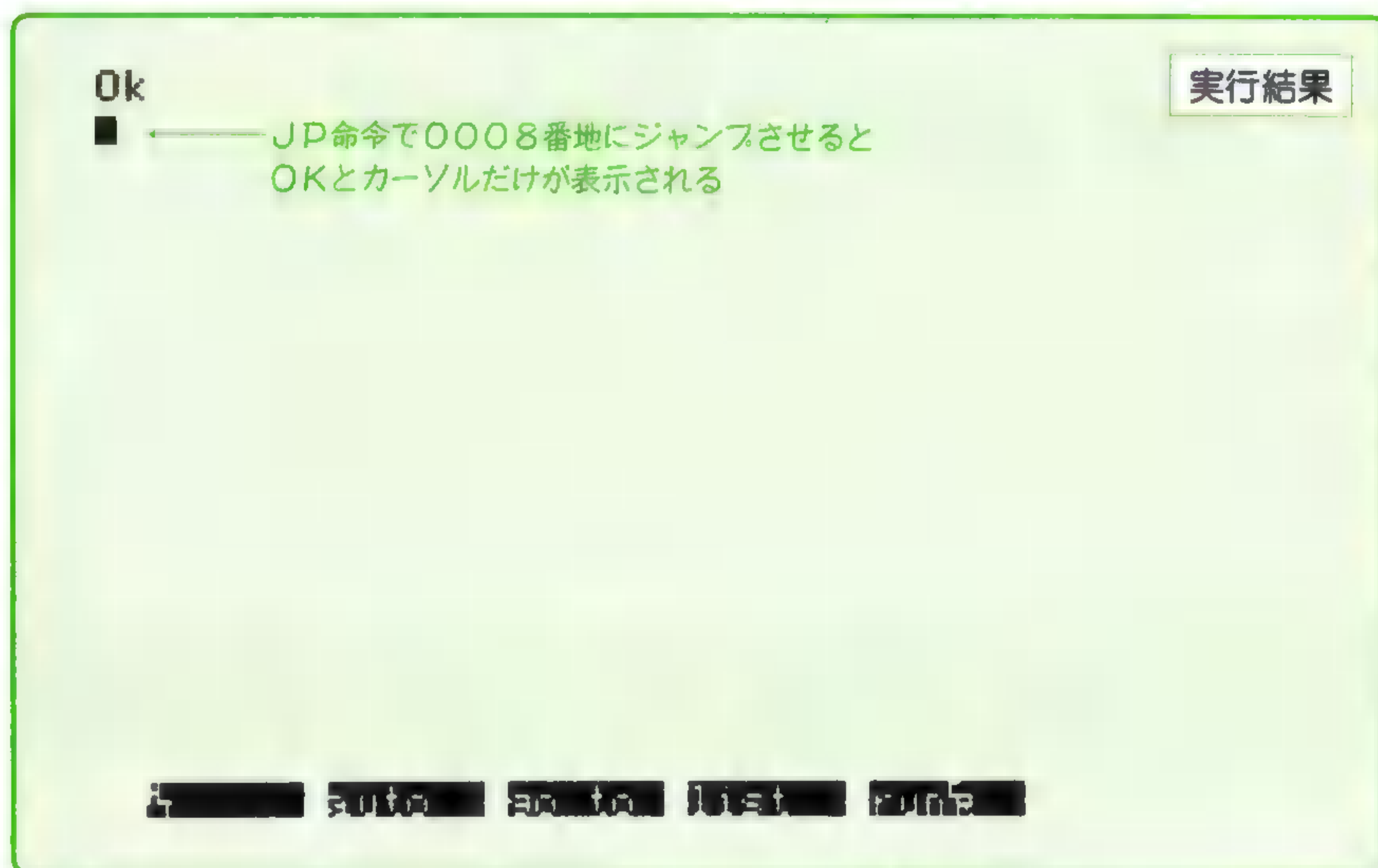
```
D000 3E EA 32 E4 F6 C3 08 00
```

ベーシックのプログラムとマシン語のプログラムを入力したら、**G D 0 0 0** (**RET**キー) で、マシン語のプログラムを実行させます。マシン語のプログラムの入力の仕方は、**16**頁を参照してください。

このマシン語のプログラムの終わりは、

```
C 3 0 8 0 0 (J P 0 0 0 8 H)
```

で、**J P**命令で**0 0 0 8**番地にジャンプさせていますから、マシン語のプログラムを実行させると、画面はつぎの実行結果に示すようになります。



この場合、ベーシックのプログラムは消滅していないので、**LIST**を入力すると、行番号**10**のベーシックステートメントが画面に表示されます。

おなじように、行番号**10**のベーシックステートメントと、まえの頁の**J P**命令で**0 0 0 0**番地にジャンプするマシン語のプログラムを入力して、マシン語のプログラムを実行させます。マシン語のプログラムを実行すると、画面はまえの頁の実行結果に示したようになります。この場合は、**LIST**を入力しても、行番号**10**のベーシックステートメントは消滅していて表示されません。

画面の左から右へ複数個の キャラクタを表示する

基本的な加算命令 INC

ここでは、INC命令を説明するために、INC命令を使って、画面の左から右に複数個のキャラクタを表示させることにします。

109頁と116頁で、LD命令を使って、画面にハートとダイヤのキャラクタを表示しました。その場合は、まずLD命令で、AレジスタにハートのキャラクタコードE9H、またはダイヤのキャラクタコードEAHを代入し、つぎに、VRAMのメモリ番地を指定して、LD命令で、指定したVRAMのメモリ番地にAレジスタのE9H、またはEAHを代入して、画面に表示させました。

ハートを画面の左上に表示

```
LD  A, E9H
LD  (F300H), A
HALT
```

ダイヤを画面の真中に表示

```
LD  A, EAH
LD  (F6E4H), A
HALT
```

ハートやダイヤなどのキャラクタを複数個、画面に表示する場合も、LD命令を使って表示することができます。

たとえば、5個のハートを表示させるには、

アセンブリ言語

```
LD  A, E9H
LD  (F300H), A
LD  (F302H), A
LD  (F304H), A
LD  (F306H), A
LD  (F308H), A
HALT
```

マシン語

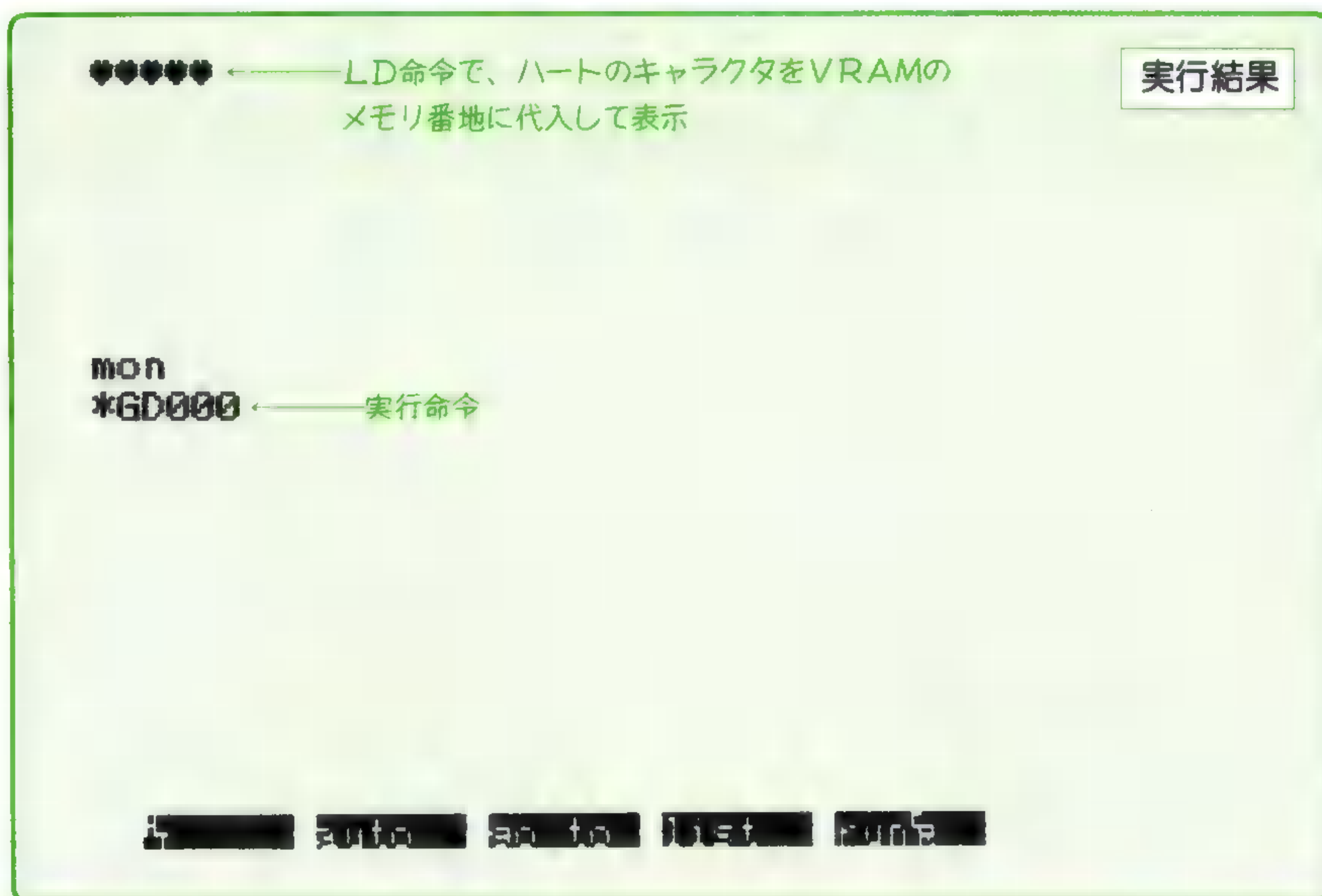
```
3E  E9
32  00  F3
32  02  F3
32  04  F3
32  06  F3
32  08  F3
76
```

というように、まずAレジスタに、ハートのキャラクタコードE9Hを代入します。そして、5個のハートを表示するVRAMのメモリ番地をひとつひとつ指定して、LD命令で、指定したVRAMのメモリ番地にひとつひとつAレジ

スタのE9Hを代入してやればよいわけです。

この場合は、ハートの表示を横40文字表示で行っています。したがって、VRAMのメモリ番地は、ひとつ置きになります。

つぎに、このプログラムの実行結果を示します。



さて、ベーシックで複数個のキャラクタを表示する場合、PRINT命令をいくつも並べてキャラクタを表示するというようなことはしないで、つぎのように、FOR～NEXT命令を使って、必要なだけPRINT命令を繰り返えして表示させます。

```
10 FOR I=1 TO 5
20 PRINT "♥";
30 NEXT I
40 END
```

マシン語にもDJNZ（139頁参照）という、ベーシックのFOR～NEXT命令とおなじ繰り返えし命令があります。このDJNZ命令を使ってキャラクタを表示するときに必要になるのが、INC命令です。もちろん、INC命令は加算命令の基本的な命令ですから、いろいろな場合に使う命令であることはいうまでもありません。

INC命令は、+1する命令

INC命令は、インクリメント命令といって、レジスタやメモリに記憶されている内容に1を足す命令で、つぎのような形をしています。

INC X

Xのところには、レジスタが置かれます。キャラクタを画面の左から右へ表示する場合、HLレジスタを使いますから、まず、HLレジスタを使って説明することにします。

さてこの場合は、HLレジスタを使いますから、INC命令のあとにあるXには、つぎに示すように、HLレジスタが置かれます。HLレジスタについては、あとで説明します。

INC HL

このHLレジスタに、F300Hが記憶されているとします。

HLレジスタに記憶されている内容=F300H

つぎに、さきに示したINC HL命令を実行させます。

INC HL

すると、HLレジスタに記憶されている内容F300Hに、1がプラスされます。

F300H+1

したがって、INC HL命令を実行したあとのHLレジスタの内容は、

HLレジスタに記憶されている内容=F301H

F301Hになります。もう一度、INC HL命令を実行させると、

INC HL

こんどは、HLレジスタに記憶されている内容は、

HLレジスタに記憶されている内容=F301H

F301Hですから、F301Hに1がプラスされます。

F301H+1

したがって、2回目のINC HLを実行したあとのHLレジスタに記憶されている内容は、

HLレジスタに記憶されている内容=F302H

F302Hになります。

HLレジスタは、あとで説明するように16ビットレジスタですから、F300Hというように4桁の16進数を扱います。これに対して、B、C、Dというレジスタは、8ビットレジスタですから、2桁の16進数を扱いますが、INC命令の働きは、HLレジスタの場合とおなじです。

ここでは、Bレジスタを取り上げて説明することにします。この場合は、Bレジスタですから、INC XのXのところに、Bレジスタを置きます。

INC B

このBレジスタに、09Hが記憶されているとしましょう。

Bレジスタに記憶されている内容=09H

つぎに、INC B命令を実行させます。

INC B

すると、Bレジスタの内容09Hに、1がプラスされます。

09H+1

したがって、INC B命令を実行したあとのBレジスタの内容は、

Bレジスタに記憶されている内容=0AH

0AHとなります。これが、INC命令の働きです。

INC命令を使って、横断線のキャラクターを表示する

では、INC命令を使ってハートを左から右に5個表示させることにします。さきに、アセンブリ言語で書いたプログラムを示します。

```
LD B, E9H ← ハートのキャラクタコード E9HをBレジスタに代入
LD HL, F300H
LD (HL), B ← Bレジスタの内容をHLレジスタに代入
INC HL
LD (HL), B
INC HL
LD (HL), B
INC HL
LD (HL), B
INC HL
LD (HL), B
JP 5C66H ← 5C66番地にジャンプ
```

HLレジスタの内容を+1する

つぎは、アセンブリ言語の説明です。1行目の、

LD B, E9H

は、LD命令でBレジスタに、ハートのキャラクタコードE9Hを代入しています。2行目の、

LD HL, F300H

で、ハートを最初に表示するVRAMのメモリ番地F300Hを、HLレジスタに代入しています。

HLレジスタは、HとLレジスタを組み合わせたもの

110頁で、CPUのなかにはAレジスタのほかに、B、C、D、E、H、Lのレジスタがあると説明しました。B、C、D、E、H、Lのレジスタは、ひとつひとつの場合、Aレジスタとおなじように8ビットレジスタで、8ビット（1バイト）しか扱うことができません。

B C D E H L ← ひとつひとつのときは8ビットレジスタ

ところが、このB、C、D、E、H、Lのレジスタは、2つずつ組み合わせて使うことができます。組み合わせの仕方は

BC DE HL ← 2つずつ組み合わせて使うことができる

BレジスタとCレジスタ、DレジスタとEレジスタ、HレジスタとLレジスタと決まっています。このようにレジスタを組み合わせて使うことを、**レジスタペア**といいます。レジスタペアにすると、16ビット（2バイト）を扱うことができるようになります。

この場合、HLレジスタを使っているのは、F300Hというように、16ビットを扱わなければならないからです。また、このことが大切なことなのですが、16ビットを扱うということだけなら、なにもHLレジスタでなくても、BCレジスタ、DEレジスタであってもよいわけです。ところが、BCレジスタ、DEレジスタを使うと、つぎに示すように、Aレジスタに記憶させたものしか使えないという問題があるのです。

LD (BC), A LD (DE), A ← Aレジスタに記憶させたものしか使えない

これに対して、HLレジスタを使うと、A、B、C、Dなどのレジスタに記憶させたものすべてが使えるようになります。

LD (HL), B LD (HL), C
LD (HL), D LD (HL), E ← A、B、C、Dなど、いずれのレジスタに記憶させても使える

このようなことから、HLレジスタペアを使うわけです。

ハートの表示

さて、説明をもとに戻して、3行目の、

LD (HL), B

のカッコのなかのHLは、2行目でHLレジスタにVRAMのメモリ番地F300Hを代入しましたからF300H。またBは、1行目で、BレジスタにハートのキャラクタコードE9Hを代入しましたからE9H。したがって、つぎのようになります。

LD (F300H), E9H

LD命令で、ハートのキャラクタコードE9Hを、VRAMのメモリ番地F300Hに代入するということは、VRAMのメモリ番地F300Hにハートを表示するということですから、1個目のハートが画面の一番左端上に表示されることになります。

つぎは、2個目のハートの表示です。画面の左から右へ表示していく場合VRAMのメモリ番地はひとつずつ増えていきます。したがってVRAMのメモリ番地にしたがってHLレジスタに記憶されている内容をひとつずつ増やしていかなければなりません。そこでINC命令を使って、HLレジスタに記憶されている内容に1をプラスします。それが、4行目の、

INC HL

です。INC HL命令を実行すると、

F300H + 1

が行われて、HLレジスタに記憶されている内容は、

HLレジスタに記憶されている内容 = F301H

となります。4行目のINC HL命令で、1プラスされたHLレジスタの内容F301Hが、5行目の、

LD (HL), B

のカッコのなかのHLの内容になります。したがって、

LD (F301H), E9H

となりますから、2個目のハートが、VRAMのメモリ番地F301Hに表示されることになります。あとはおなじことの繰り返しなので、説明は省略します。

アセンブリ言語をマシン語になおす

では、アセンブルです。巻末のZ-80アセンブル表を使ってアセンブリ言語をマシン語に直します。アセンブルの仕方は108頁や113頁で説明したとおりです。したがって、アセンブリ言語とマシン語を並べて示します。

アセンブリ言語	マシン
LD B, E9H	06 E9
LD HL, F300H	21 00 F3
LD (HL), B	70
INC HL	23
LD (HL), B	70
INC HL	23
LD (HL), B	70
INC HL	23
LD (HL), B	70
INC HL	23
LD (HL), B	70
JP 5C66H	C3 66 5C

このマシン語のプログラムを、D000番地から入力して実行させると、つぎの実行結果に示すように、ハートが3個しか表示されません。これは、横40文字表示で実行しているからです。

*** ← 横40文字表示なので、
3個しか表示されない

実行結果

mon
*GD000 ← 実行命令

*■ ← JP 5C66Hでマシン語のモニタにジャンプして表示

h 0000 0000 0000 0000 0000 0000 0000 0000

さきに説明したように、横40文字表示の場合は、VRAMのメモリ番地は、ひとつ置きにしか使うことができません。したがって、INC HL命令で、HLレジスタに記憶されている内容にプラス1する場合、

```
INC HL ← INC命令を1回実行する
LD (HL), B ← HLの内容が1プラスされる
```

カッコのなかのHLの内容が、表示できないVRAMのメモリ番地になるということが起こります。そこで、ハートが、3個しか画面に表示されないわけです。

この場合は、ベーシックのコマンド・レベルでWIDTH命令を使って、

```
WIDTH 80, 25
```

と入力して、画面表示を横80文字表示に変更すると、つぎの実行結果に示すように、このままでもハートが5個、画面に表示されます。

実行結果

***** ← WIDTH命令で画面表示を80文字にするとハートを5表示する

200
110000 ← 実行命令

横 ← JP 5C66Hで表示

ただし、横80文字表示でキャラクタを表示させると、表示が非常に小さいので見にくくなります。

そこで、横40文字表示で、キャラクタを表示するにはどうしたらよいのかが問題となります。横40文字表示で、キャラクタを表示するには、つぎに説明するようにすればよいのです。

130

横40文字表示で、ハートを5個表示するには

横40文字表示のVRAMのメモリ番地は、巻末のメモリ番地表を見るとわかるように、F300HのつぎはF302H、そのつぎはF304Hとなっています。したがって横40文字表示で、ハートを画面の左から右へ5個表示するには、HLレジスタに記憶されている内容に2をプラスすればよい、ということになります。

INC HL命令を実行すると、HLレジスタに記憶されている内容に1プラスするわけですから、HLに記憶されている内容に2をプラスするには、つぎに示すように、

```
INC HL
INC HL
```

← INC命令を2回実行させる

INC HL命令を2つ重ねて、2回実行させればよいということになります。

したがって、さきのプログラムの4行、6行、8行、10行目にあるINC HL命令に、もうひとつINC HL命令を追加して、INC HL命令を2つにして実行させると、つぎの実行結果に示すように、5個のハートが表示されます。

実行結果

***** ← INC HL命令を2個にすると
5個のハートを表示

MON
*6D000 ← 実行命令

本題 ← JP 5C66Hで表示

auto auto list auto

06	E9	
21	00	F3
70		
23	23	
70		
23	23	
70		
23	23	
70		
23	23	
70		
C3	66	5C

画面の右から左へ複数個の キャラクタを表示する

基本的な減算命令 DEC

123頁で基本的な加算命令であるINC命令について説明しました。INC命令に対して、基本的な減算命令としてDEC命令があります。ここでは、DEC命令を説明するために、DEC命令を使って、画面の右から左へ複数個のキャラクタを表示することにします。

DEC命令は、－1する命令

DEC命令は、デクリメント命令といって、INC命令の逆の働きをする命令です。つまり、レジスタに記憶されている内容から、1を引く働きです。DEC命令は、つぎのような形をしています。

DEC X

DEC XのXのところには、レジスタが置かれます。

キャラクタを画面の右から左へ表示する場合、HLレジスタを使って行いますから、まず、HLレジスタを使って、DEC命令の働きを説明することにします。したがって、DEC命令のあとにあるXには、つぎに示すように、HLレジスタが置かれることになります。

DEC HL

この場合、HLレジスタを使うのは、127頁のINC命令で、HLレジスタを使ったのとおなじ理由によります。HLレジスタ、およびHLレジスタを使う理由については、INC命令の項で説明していますから参照してください。

さてHLレジスタに、F696Hが記憶されているとします。HレジスタとLレジスタが2つ組み合わされてHLレジスタとなると、16ビット（2バイト）を扱うレジスタになります。したがってF696Hというように、4桁の16進数を扱います。

HLレジスタに記憶されている内容＝F696H

画面の右から左へ複数個のキャラクタを表示する

つぎに、DEC HL命令を実行させます。

DEC HL

すると、HLレジスタに記憶されている内容F696Hから1が引かれます。

F696H-1

したがって、DEC HL命令を実行したあとのHLレジスタの内容は、

HLレジスタに記憶されている内容=F695H

F695Hとなります。もう一度、DEC HL命令を実行させると、

DEC HL

こんどは、HLレジスタに記憶されている内容はF695Hですから、F695Hから1が引かれます。

F695H-1

したがって、2回目のDEC HL命令を実行したあとのHLレジスタの内容は、

HLレジスタに記憶されている内容=F694H

F694Hとなります。

B、C、Dといった、8ビットレジスタに記憶されている内容から1を引く場合も、HLレジスタの場合とおなじです。8ビットレジスタの場合は、2桁の16進数を扱うことになります。

ではつぎに、Bレジスタを取りあげて、DEC命令の働きをみてみることにします。

Bレジスタの場合は、DEC XのXのところに、Bレジスタを置きます。

DEC B

このBレジスタに、28Hが記憶されているとします。

Bレジスタに記憶されている内容=28H

つぎにDEC B命令を実行させます。

DEC B

すると、Bレジスタに記憶されている内容28Hから、1が引かれます。

28H-1

したがって、DEC B命令を実行したあとのBレジスタの内容は、

Bレジスタに記憶されている内容=27H

27Hになります。

これがDEC命令の働きです。

DEC命令を使って、複数のキャラクタを表示する

では、DEC命令を使って、複数のキャラクタを表示させることにします。
この場合は、○を5個表示することにします。

つぎに、アセンブリ言語で書いたプログラムを示します。

```
LD  C, EDH      ← ○のキャラクタコードEDHをCレジスタに代入
LD  HL, F34EH   ← HLレジスタにVRAMのメモリ番地F34EHを代入
LD  (HL), C     ← Cレジスタの内容をHLレジスタに代入
DEC  HL         ← HLレジスタの内容から1を引く
LD  (HL), C
DEC  HL
LD  (HL), C
DEC  HL
LD  (HL), C
DEC  HL
LD  (HL), C
JP  5C66H
```

では、アセンブリ言語のプログラムの説明です。1行目の、

LD C, EDH

はLD命令で、Cレジスタに○のキャラクタコードEDHを代入しています。
2行目の、

LD HL, F34EH

では、LD命令で○を最初に表示するVRAMのメモリ番地F34EHを、HLレジスタに代入しています。メモリ番地F34EHは、画面の右端上の位置になります。3行目の、

LD (HL), C

のカッコのなかのHLは、2行目でHLレジスタにVRAMのメモリ番地F34EHを代入していますから、F34EHとなります。また、Cは、1行目でCレジスタに○のキャラクタコードEDHを代入しましたから、EDHとなります。したがって、3行目は、つぎのようになります。

LD (F34EH), EDH

LD命令で、○のキャラクタコードEDHを、VRAMのメモリ番地F34E

画面の右から左へ複数個のキャラクタを表示する

Hに代入するということは、VRAMのメモリ番地F34EHに○を表示するということです。画面の一番右端に1個目の○が表示されます。

つぎは、2個目の○の表示です。画面の右から左へ○を表示していく場合、VRAMのメモリ番地は、ひとつずつ小さくなっていきます。したがってVRAMのメモリ番地にしたがって、HLレジスタに記憶されている内容をひとつずつ減らしていかなければなりません。そこで、DEC命令を使って、HLレジスタの内容から1を引きます。それが4行目の、

DEC HL

です。DEC HL命令を実行すると、

F34EH-1

が行われて、HLレジスタに記憶されている内容は、

HLレジスタに記憶されている内容=F34DH

となります。4行目のDEC HL命令で、マイナス1されたHLレジスタの内容F34DHが、5行目の、

LD (HL), C

カッコのなかのHLの内容になります。したがって、

LD (F34DH), EDH

となりますから、VRAMのメモリ番地F34DHの位置、つまり、最初に表示された○のひとつ左側の位置に、2個目の○が表示されることになります。

あとは、これまで説明したように、

DEC HL

で、それまでHLに記憶されていた内容から1を引き、

LD (HL), C

で、1引かれたHLに記憶されている内容に○を表示するという繰り返しです。すから、説明は省略します。最後の行の、

JP 5C66H

は、マシン語モニタへのジャンプです。したがって、ふたたび、マシン語のコマンドを受けつける*印を表示します。

アセンブリ言語をマシン語になおす

では、アセンブルです。巻末のZ-80アセンブル表を使ってアセンブリ言語をマシン語に直します。アセンブルの仕方は、113頁で説明したとおりです。したがって、アセンブリ言語とマシン語を並べて示します。

アセンブリ言語	マシン語
LD C, EDH	0E ED
LD HL, F34EH	21 4E F3
LD (HL), C	71
DEC HL	2B
LD (HL), C	71
DEC HL	2B
LD (HL), C	71
DEC HL	2B
LD (HL), C	71
DEC HL	2B
LD (HL), C	71
JP 5C66H	C3 66 5C

つぎに、このプログラムのD000番地から入力して実行した結果を示します。

実行結果

〇を3個しか表示しない

000

MON

*6D000 ← 実行命令

*■ ← JP 5C66Hで表示

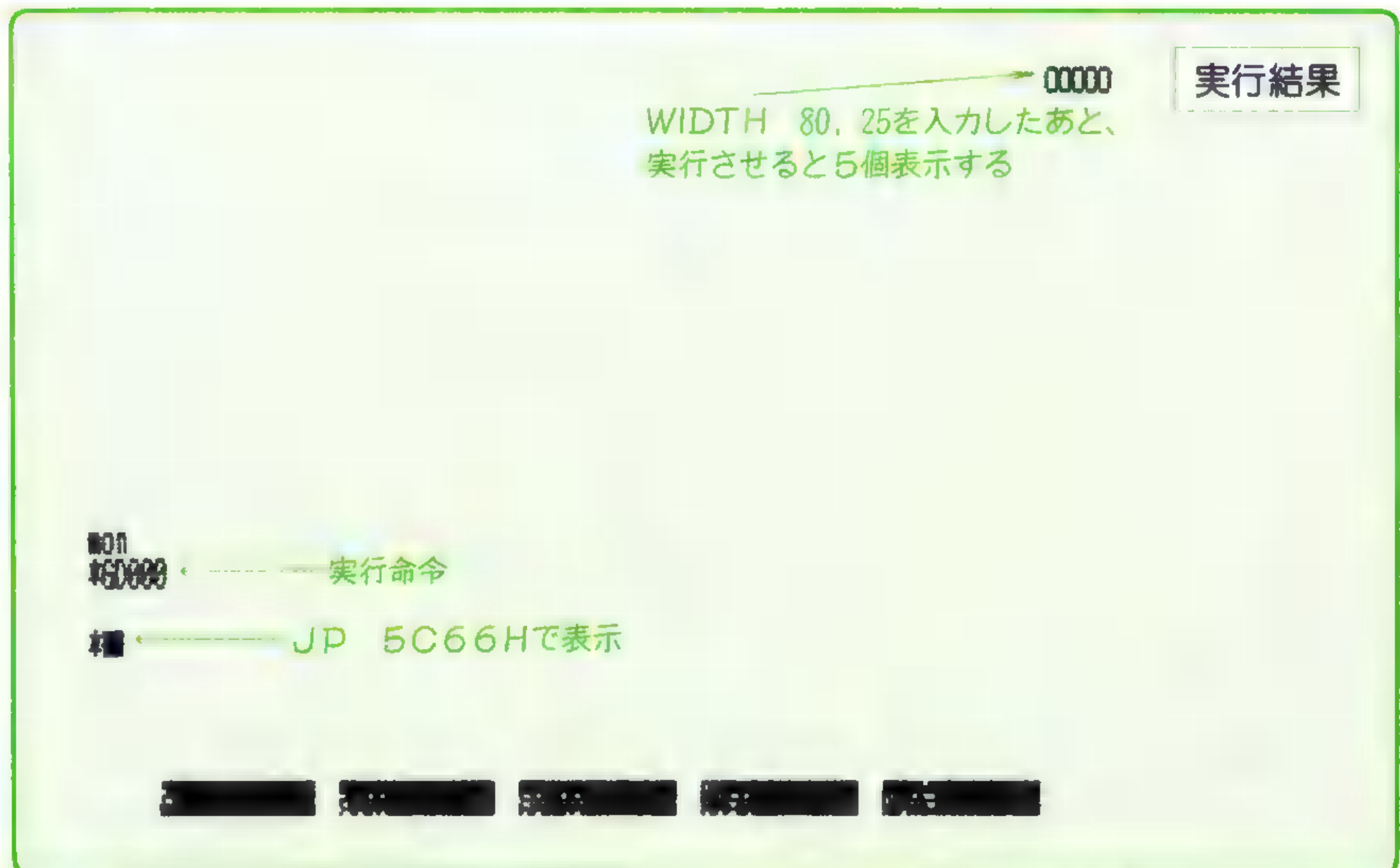
L auto on to list run

画面の右から左へ複数個のキャラクタを表示する

ところで、○を5個表示するはずのプログラムが、実行結果では、○を3個しか表示していません。これは、横40文字で実行させているからです。横40文字表示の場合は、VRAMのメモリ番地は、ひとつ置きに使われるので、このような結果になるのです。そこで、つぎのように、WIDTH命令で、

WIDTH 80, 25 ← ベーシックのコマンド・レベルに戻し、画面表示を変更する

横80文字表示に変更して実行すると、○が5個表示されます。



横40文字表示で、○を5個表示するには

ただ、横80文字表示では○が小さくて、みにくいということがあります。そこで、横40文字表示で、○を5個表示することにします。

VRAMのメモリ番地は巻末のメモリ番地表をみるとわかるように、横40文字表示の場合、右から左へF34EH、F34CH、F34AH、F348Hとふられています。したがって、○を右から左へ表示するには、HLに記憶されている内容を、2ずつ引いていけばよいということになります。

DEC HL命令は、HLに記憶されている内容から、1を引くわけですから、HLに記憶されている内容から2を引くには、つぎに示すように、

DEC HL | **DEC HL** ← DEC HL命令を2回実行させる

DEC HL命令を2個重ねて、2回実行すればよいということになります。

つぎに、まえのプログラムの4行目、6行目、8行目、10行目のDEC HL命令のあとに、それぞれDEC HL命令を追加したプログラムを、ダンプリストで示します。

```
D000 0E ED 21 4E F3 71 2B 2B
D008 71 2B 2B 71 2B 2B 71 2B
D010 2B 71 C3 66 5C FF FF FF
```

つぎに示すのが、DEC HL命令を2個にしたプログラムの実行結果です。

00000

DEC HL命令を2
重ねて、実行させる

実行結果

mon

*GD000 ← 実行命令

* ← JP 5C66Hで表示

5 2410 2010 1131 0000

引く数だけDEC HL命令を重ねる

このように、HLに記憶されている内容から2を引くときは、DEC HL命令を2個重ねて、2度実行させればよいのです。したがって、HLに記憶されている内容から3を引くときは、つぎのように、

```
DEC HL }
DEC HL } 3を引くときは、3重ねる
DEC HL }
```

DEC HL命令を3個重ねて引くということになります。

ベーシックのFOR~NEXT命令とおなじ繰り返し処理をさせるには

繰り返し処理専門の命令 DJNZ

123頁ではLD命令とINC命令を使って、また、132頁ではDEC命令を使って、キャラクタを画面に表示しました。LD命令、INC命令、DEC命令を使ってキャラクタを表示したときは、わずか5個のキャラクタの表示だったからいいようなものの、これが40も50もといった、数多くのキャラクタの表示だったら、プログラムが非常に長いものになってしまいます。

なぜなら、LD命令、INC命令、DEC命令で、画面にキャラクタを表示する方法は、つぎに示すように、ベーシックのLOCATE命令とPRINT命令で、キャラクタを画面に表示するのと、まったくおなじ方法だからです。

```
10 LOCATE 0,0  
20 PRINT "♥"  
30 LOCATE 1,0  
40 PRINT "♥"  
50 LOCATE 2,0  
60 PRINT "♥"
```

LOCATE文で、列番号、行番号を指定して
その位置にハートを表示する

ベーシックではこのような場合、つぎに示すように、

```
10 FOR I=1 TO 40  
20 PRINT "♥";  
30 NEXT I  
40 END
```

FOR~NEXT命令で繰り返しして処理する

繰り返しし処理の命令である、FOR~NEXT命令を使って処理します。

マシン語でも、ベーシックのFOR~NEXT命令とおなじように、繰り返しし処理をする命令があります。**DJNZ命令**です。したがって、繰り返しし処理をするときは、DJNZ命令を使って処理します。

DJNZ命令は、つぎのような形をしています。

DJNZ

DJNZのマシン語=10 e-2

繰り返えしの命令DJNZのマシン語の10のあとにあるe-2については、あとで説明します。ここでは、e-2には、繰り返えし処理をするスタート番地を相対アドレスで書く、とだけおぼえておいてください。

繰り返えし処理をする回数はBレジスタにセット

ところで、DJNZ命令をみると、繰り返えし処理をするスタート番地の指定をするところがありますが、何回繰り返えして処理をするのか、繰り返えす回数を指定するところがありません。これは、繰り返えす回数は、Bレジスタにセットすると決まっているからです。

Bレジスタ=繰り返えしの回数をセット

ベーシックのFOR~NEXT命令では、FOR命令に繰り返えしの回数を書きます。たとえば、40回繰り返えすときは、

FOR I=1 TO 40 初期値を1、終値を40とする

それとおなじように、マシン語の場合は、Bレジスタに繰り返えしの回数をセットするわけです。たとえば、40回繰り返えし処理するときは、

LD B, 28H

というようにです。LD B, 28Hは、Bレジスタに28Hを代入するということです。28Hは、10進数で40です。10進数を16進数に変換するときは、巻末に示した10進数↔16進数変換表を使って変換してください。

繰り返えし処理する命令は、BレジスタとDJNZ命令の間に置く

ところで、ベーシックのFOR~NEXT命令は、FOR命令に繰り返えし処理をする回数をセットして、繰り返えし処理するステートメントを、つぎに示すように、FOR命令とNEXT命令の間に置きます。

FOR I=1 TO 40

繰り返えし処理をするステートメント

NEXT I

ベーシックのFOR~NEXT命令とおなじ繰り返えし処理をさせるには

DJNZ 命令の場合もおなじように、繰り返えし処理をする命令を、つぎに示すように、繰り返えす回数をセットしたBレジスタと、DJNZ 命令との間に置きます。

Bレジスタ

繰り返えし処理をする命令

DJNZ

これが、繰り返えし処理するDJNZ命令の形です。

40個の●を画面に表示する

DJNZ 命令の形がわかったところで、画面の左から右へ40個の●を表示させて、具体的にDJNZ命令の使い方について説明していくことにします。

DJNZ 命令を使って40個の●を表示するプログラムを、おおまかな図で示すと、つぎのようになります。

Bレジスタに繰り返えす回数40をセット

繰り返えし処理をする命令、つまり●を表示する命令

DJNZ

画面にキャラクタを表示する方法は、109頁で説明しました。まず、●のキャラクタコードECHをAレジスタに代入します。

LD A, ECH

となります。つぎは、表示するVRAMのメモリ番地をHLレジスタに代入します。この場合は画面の左から右へ●を表示するわけですから、HLレジスタには、画面の一番左端上のVRAMのメモリ番地F300Hを代入します。

LD HL, F300H

それから、Aレジスタに代入したECHを、VRAMのメモリ番地を記憶しているHLレジスタに代入します。

LD (HL), A

以上のことを、わかりやすいように、ここまでをまとめておきます。

LD A, ECH ← ●のキャラクタコードECHをAレジスタに代入

LD HL, F300H ← VRAMのメモリ番地F300HをHLレジスタに代入

LD (HL), A ← Aレジスタの内容をHLレジスタに代入

そのほかに、HLレジスタに記憶されている内容に+1して、メモリ番地をひとつひとつ増やしていくINC命令がありますが、このINC命令をどこに置くかが問題なので、ひとまずINC命令を除いて説明します。

さて、上にまとめた命令のうち、どの命令をBレジスタとDJNZ命令の間に置けばよいかです。キャラクタコードECHは、一度Aレジスタに代入すればよいわけです。最初のVRAMのメモリ番地F300Hも、一度HLレジスタに代入すればよいわけです。

したがって、■を画面に表示していくには、LD (HL), A命令を、BレジスタとDJNZ命令の間に置いて繰り返えし処理すればよいということになります。Bレジスタに、繰り返えす回数40をセットする命令は、さきに取りあげたようにLD B, 28Hですから、

LD A, ECH	← ECHをAレジスタに代入
LD HL, F300H	← F300HをHLレジスタに代入
LD B, 28H	← 繰り返えしの回数40をBレジスタに代入
LD (HL), A	← ECHをHLレジスタに代入して画面に表示
DJNZ	← 繰り返えし処理をするスタート番地へ戻す

となります。

このプログラムをマシン語になおして実行しても、VRAMのメモリ番地F300H、つまり画面の左端上で■が繰り返えし表示されるだけです。なぜなら、HLレジスタに記憶されている内容がF300Hのまま変化しないからです。そこでINC命令で、HLレジスタに記憶されている内容に+1します。

HLレジスタに記憶されている内容に+1する命令は、123頁のINC命令で説明しているように、

INC HL

となります。

このINC HL命令は、LD (HL), A命令とDJNZ命令の間に入れます。なぜなら、INC HL命令も繰り返えさないと、1、2、3というように数が増えていかないからです。

ただしこの場合は、横40文字表示で■を表示していく関係から、つぎに示すように、INC HL命令を2個使います。この理由は131頁で説明していますので、その頁を参照してください。

ベーシックのFOR～NEXT命令とおなじ繰り返し処理をさせるには

```
LD B, 28H
```

```
LD (HL), A
```

```
INC HL
```

```
INC HL
```

```
DJNZ
```

INC HL命令を2個、この間に入れる

このようにすると、まず、HLレジスタに代入されているF300Hで、LD (HL), Aは、つぎのようになって、

```
LD (F300H), ECH
```

1個目の●を画面の左端上に表示します。そして、最初のINC HL命令で、HLレジスタに記憶されている内容F300Hに+1します。したがって、HLレジスタに記憶されている内容は、

HLレジスタに記憶されている内容=F301H

となります。つぎのINC HL命令で、HLレジスタに記憶されている内容F301Hに+1しますから、HLレジスタに記憶されている内容は、

HLレジスタに記憶されている内容=F302H

となります。

INC HL命令を2回実行したあと、DJNZ命令で、繰り返し処理をするスタート番地に戻りますから、HLレジスタに記憶されている内容F302Hで、LD (HL), Aは、つぎのようになります。

```
LD (F302H), ECH
```

つまり、2個目の●が、VRAMのメモリ番地F302Hに表示されることになります。つまり、画面に表示されている1個目の●の、つぎの位置に表示されるということです。

このことを、Bレジスタにセットした40まで繰り返しして、●を表示していくわけです。Bレジスタにセットした40が0になると、DJNZ命令による繰り返しをやめて、つぎのメモリ番地に記憶されている、命令の実行に移ります。

なぜ、Bレジスタにセットした40が0になるのかというと、DJNZ命令で繰り返し処理をするスタート番地に戻るたびに、Bレジスタにセットした40から1を引いて、そのうえでスタート番地に戻るからです。

では、アセンブルです。

アセンブリ言語をマシン語になおす

このプログラムの場合は、DJNZ命令で、繰り返えし処理を行うスタート番地に戻さなければなりません。どのメモリ番地に戻すのかを説明するにはプログラムをメモリ上に割りつけておかなければなりませんから、ここでは、メモリ番地、マシン語、それにアセンブリ言語といった順で、プログラムを示します。メモリ番地はD000Hからにします。

メモリ番地	マシン語	アセンブリ言語
D000	3E EC	LD A, ECH
D002	21 00 F3	LD HL, F300H
D005	06 28	LD B, 28H
D007	77	LD (HL), A
D008	23	INC HL
D009	23	INC HL
D00A	10 FB	DJNZ D007H
D00C	C3 66 5C	JP 5C66H

10のあとのe-2とは

DJNZ命令をマシン語になおすと、10 e-2となりますが、このe-2とはなんのことでしょうか。まえにe-2には、繰り返えし処理をするスタート番地を書くと言明しました。

10 e-2 ← 繰り返えし処理をするスタート番地

そのことに違いはありません。ただ、e-2と指定されている以上、何か特別な指定の仕方があるのではないかと思います。そのとおりで、e-2に書くメモリ番地は、**相対アドレス指定**であるということです。

うえに示したプログラムの、最後の行のJP命令をみてください。JP命令では、メモリ番地5C66Hにジャンプさせるとき、2バイトのメモリ番地をそのまま書きます。

C3 66 5C ← 5C66Hをそのまま書く

このように、2バイトのメモリ番地を、そのまま書くことを、**絶対アドレス指定**といいます。

では、下から2行目のDJNZ命令をみてください。ここでは説明の都合上e

— 2 のところにFBHを入れています。

10 FB ← FBHが戻っていく先のメモリ番地

このプログラムの場合、FBHが戻っていく先のメモリ番地ですが、メモリ番地は2バイトですから、どこにもFBHという1バイトのメモリ番地はありません。

端的に言って、これが相対アドレス指定というものです。相対アドレス指定では、FBHというように、1バイトでメモリ番地の指定を行います。

10 FB ← 1バイトでメモリ番地の指定を行う

ここでなぜ、e-2の箇所にFBHが入っているのかはあとでわかります。

相対アドレス指定のやり方

では、相対アドレス指定は、どのようにして行うのかということです。相対アドレス指定について根本から説明すると、あれやこれや説明しなければならず、非常にむずかしいものになってしまい、かえって理解できなくなってしまうと思うので、ここでは必要最少限の説明にとどめます。

CPUのなかには、PC（プログラムカウンタ）というものがあります。このPCは常に、CPUがつぎに実行する命令が記憶されているメモリ番地を指しています。たとえば、うえのプログラムを実行させるとき、

*GD000 

を入力して、RETキーを叩きます。このとき、PCは、

PC → D000H

を指します。CPUはPCが指しているD000Hをみて、D000Hのメモリ番地に記憶されている命令を取り出して実行にかけると、PCは、

PC → D002H

つぎに実行するメモリ番地D002Hを、自動的に指します。

相対アドレス指定は、このPCが指しているメモリ番地に働きかけて、メモリ番地の指定をもとに戻していくやり方です。

では、144頁のプログラムの下から2行目をみてください。10 FBを実行すると、PCは、つぎにCPUが実行する命令が記憶されている最後の行の先頭のメモリ番地D00CHを指します。

D00A 10 FB

D00C ← C3 66 5C

PC（プログラムカウンタ）は
このメモリ番地を指す

そこで、PCが指しているD00C番地を、繰り返えし処理する最初の命令が記憶されているメモリ番地まで戻してやります。CPUは、PCが指しているメモリ番地をみて、そのメモリ番地に記憶されている命令を実行しますから、PCが指しているメモリ番地を戻してやれば、CPUは、ふたたび、そのメモリ番地に記憶されている命令を、実行するというわけです。

さて、このプログラムの場合、繰り返えし処理をするスタート番地、つまり最初のメモリ番地はどこでしょうか。

メモリ番地D005HとD006Hに記憶されているマシン語06 28は、繰り返えしの回数40をBレジスタにセットする命令ですから、繰り返えす必要はありません。したがって、繰り返えし処理するスタート番地は、そのつぎの行のD007Hということになります。そこで、PCが指しているメモリ番地D00C番地を、繰り返えし処理をするスタート番地D007まで戻します。PCの指しているメモリ番地の戻し方は、つぎのように行います。

D00C	C3	←	10	FBを実行してPCが指しているメモリ番地
D00B	FB	←	-1	すると、このメモリ番地に戻る
D00A	10	←	-2	すると、このメモリ番地に戻る
D009	23	←	-3	すると、このメモリ番地に戻る
D008	23	←	-4	すると、このメモリ番地に戻る
D007	77	←	-5	すると、このメモリ番地に戻る

ということで、-5になります。したがって、DJNZ命令のマシン語10のあとのe-2のところに-5と書けばよい、ということになります。

10 -5

もちろん、-5は10進数ですから、そのまま書くわけにはいきません。-5を16進数になおすと、FBHということになります。そこで、e-2のところにFBHを置いて、

10 FB

として説明してきたわけです。

FBHは、符号つき16進数です。符号つき16進数も249頁に表にして示してありますから、その表を使って変換してください。-5の場合は、-5のあるところから横と上を見ます。横に上位ビット、上に下位ビットが示してあります。したがって、-5は上位ビットがF、下位ビットBで、FBHになるということがわかります。

ところで、実行結果をみるかぎりでは、ただ、画面に40個の●が表示されているだけですが、実際に実行させて見ると、ベーシックの場合とは比較にならない速さで表示されます。マシン語の処理の速さが、これでもわかります。

画面の上から下に向けて●を表示

16ビット（2バイト）の加算命令 ADD

139頁で、DJNZ命令とINC命令を使って、画面の左から右に向けて●を表示しました。ここでは、画面の下に向けて●を表示していくことにします。

画面の下に向けて●を表示していくとき必要になるのが、16ビット（2バイト）の加算です。16ビットの加算を行う命令は、**ADD命令**です。なお、8ビット（1バイト）の加算命令もADD命令です。

16ビットの加算と8ビットの加算の違うところは、16ビットの加算の場合、16ビットが扱えるように、レジスタをHL、BC、DEというレジスタペアにして加算を行うのに対して、8ビットの加算は、A、B、Cという8ビットレジスタのままで加算を行うということです。

つぎに、16ビットの加算命令ADDの形を示します。

ADD HL, XX

16ビットの加算命令ADDでは、うえに示したように、ADD命令のあとに、HLレジスタペアを使います。

そのあとにあるXXのところには、BC、DE、HLといったレジスタペアのうちの、いずれかひとつを選んで置きます。

そしてHLレジスタには、足される数を入れます。XXのところに置いたレジスタペアには、足す数を入れます。たとえば、つぎのような計算を行います。

3333H + 2222H =

↑ 足される数 ↑ 足す数

足される数3333Hは、HLレジスタに入れます。足す数2222Hは、BC、DE、HLのなかからBCレジスタを選んで、BCレジスタに入れることにします。

したがって、この足し算のプログラムは、つぎのようになります。

画面の上から下に向けて●を表示

```
LD HL, 3333H ← 3333HをHLレジスタに代入
LD BC, 2222H ← 2222HをBCレジスタに代入
ADD HL, BC ← ADD命令で、HLレジスタとBCレジスタの値を足す
HALT
```

加算命令ADDを実行して、HLレジスタの3333HとBCレジスタの2222Hを足すと、その結果は5555Hになります。この計算結果の5555Hは、HLレジスタに記憶されて残ります。これがADD命令を使った16ビット加算です。では、アセンブルです。

アセンブリ言語	マシン
LD HL, 3333H	21 33 33
LD BC, 2222H	01 22 22
ADD HL, BC	09
HALT	76

加算の仕方がわかったところで、●を画面の上から下に向けて表示します。

●を画面の下に向けて表示する

ところで、●を画面の下に向けて表示する場合、なぜ16ビットの加算が必要なのでしょう。それは、巻末に示したVRAMのメモリ番地表を見るとわかるように、●を表示させるVRAMのメモリ番地が、画面の下にいくにしたがって増えているからです。また、VRAMのメモリ番地がF300Hというように、16ビット（2バイト）だからです。

たとえば、最初の●を画面の上のメモリ番地F328Hに表示したとします。つぎに、その真下の位置、つまりメモリ番地F3A0Hに2個目の●を表示するには、つぎの図に示すように、最初に●を表示したメモリ番地F328Hに120（79番地+41番地=120番地）を足さないと、2個目の●を表示するメモリ番地F3A0Hを指定することができません。



そこで、16ビットの加算が必要になるのです。

では、プログラムを作ります。まず、LD命令で、画面に表示する●のキャラクタコードECHを、Aレジスタに代入します。

LD A, ECH ●のキャラクタコードECHをAレジスタに代入

この場合は、●を画面の真中を上から下に向けて表示していきますから、1個目の●を表示するVRAMのメモリ番地F328Hを、LD命令でHLレジスタに代入します。

LD HL, F328H メモリ番地F328HをHLレジスタに代入

139頁で、ベーシックのFOR～NEXT命令とおなじく、繰り返しし処理をするDJNZ命令について説明しました。このDJNZ命令を使うと、プログラムが短くてすみますから、ここでもDJNZ命令を使います。

139頁で説明したように、DJNZ命令で繰り返しし処理をする場合、繰り返す回数は、Bレジスタにセットします。つまり、画面の上から下に向けて16個の●を表示させるとしたら、Bレジスタに16をセットするわけです。16は10進数ですから、16進数になおします。16を16進数になおすと10Hになります。この10HをLD命令を使って、Bレジスタにセットします。

LD B, 10H 繰り返す回数をBレジスタにセット

繰り返す回数をBレジスタにセットしたら、つぎに、繰り返しし処理する命令を置いていきます。この場合は、まず●を表示する命令です。●のキャラクタコードECHは、Aレジスタに代入しました。画面に●を表示するには、Aレジスタに代入したキャラクタコードECHを、VRAMのアドレスを指定して、そこにLD命令で代入してやればよいわけです。この命令は、つぎのようになります。

LD (HL), A Aレジスタに代入したキャラクタコードをVRAMのメモリ番地を指定して代入

AEO命令で、メモリ番地に120を加算

さて131頁で、画面の左から右へ●を表示していくとき、INC命令を2回実行させ、HLに記憶されている内容（メモリ番地）に2を足し、その値をLD(HL), AのなかのカッコのなかのHLに送り、カッコのなかのHLの値を変化させて表示していきしました。

画面の左から右への表示の場合は、HLレジスタに記憶されている内容（メモリ番地）に2ずつ足すことで、INC命令を使うことができたが、この場

合は、さきに説明したように、HLレジスタに記憶されている内容に120を足さないと、つぎに表示する位置の指定ができません。そこで、ADD命令を使うわけです。この場合の加算は、

メモリ番地+120

です。この加算を行うには、さきに説明したように、それぞれの値を16ビットのレジスタに代入しなければなりません。

メモリ番地は足される数なので、HLレジスタに代入します。120は、足す数なので、BCあるいはDEレジスタに代入することになります。

ところで、足される数であるメモリ番地は、すでに2行目でHLレジスタに代入していますから、新たにHLレジスタに代入する必要はありません。足す数である120を、BC、DEレジスタのいずれかへ代入すればよいわけです。ここでは120を、DEレジスタに代入することにします。もちろん、120は10進数ですから、そのままDEレジスタに代入することはできません。120を16進数になおすと、

120=78H

となります。78Hは、8ビット（1バイト）の数です。8ビットの数78Hを、16ビットのDEレジスタに代入するには78Hのまえに00をつけて、

78H=0078H・———00をつけて16ビットにする

というように、16ビットの数にして代入します。

LD DE, 0078H・———120をDEレジスタに代入

メモリ番地に足す120をDEレジスタに代入したら、HLレジスタのメモリ番地と、DEレジスタの120をADD命令で加算します。

ADD HL, DE・———HLレジスタの値とDEレジスタの値を足す

ADD命令が実行されて加算された結果は、HLレジスタに記憶されます。

つまり、2行目でHLレジスタに代入したメモリ番地F328Hに、DEレジスタに代入した0078Hを、ADD命令で加算したとすると、

F328H+0078H=F3A0H・———HLレジスタに記憶される

加算した結果のF3A0Hは、HLレジスタに記憶されるというわけです。

ここで、巻末に示したVRAMのメモリ番地表をみてください。ADD命令で算出されたF3A0Hが、2行目でHLレジスタに代入したメモリ番地F328Hの真下にあることがわかんと思います。

さて、ADD命令で加算した結果の値を、LD (HL), AのカッコのなかのH

Lに送って、その位置に●を表示させます。

このことを、Bレジスタにセットした16回繰り返えさせます。そうすることで●を画面の上から下に向けて表示させて行くことができます。

このことを16回繰り返えさせるためには、DJNZ命令を使って、繰り返えし処理する最初の命令LD (HL), Aが記憶されているメモリ番地に戻さなくてはなりません。

DJNZ命令で戻るべきメモリ番地を指定

DJNZ命令で繰り返えし処理をするには、繰り返えし処理をする最初の命令が記憶されているメモリ番地に戻します。この場合は、LD (HL), Aが記憶されているメモリ番地です。ただし、DJNZ命令の場合、繰り返えし処理をするために戻すメモリ番地の指定は、2バイトのメモリ番地を直接指定する絶対アドレス指定ではなく、1バイトで指定する相対アドレス指定です(145頁参照)。したがって、プログラムをメモリ上に割り当てないと、DJNZ命令で戻るメモリ番地の指定ができませんので、これまで説明してきたプログラムを、メモリ番地C000番地から割り当てることにします。

メモリ番地	マシン語	アセンブリ言語
C000	3E EC	LD A, ECH
C002	21 28 F3	LD HL, F328H
C005	06 10	LD B, 10H
C007	11 78 00	LD DE 0078H
C00A	77	LD (HL), A
C00B	19	ADD HL, DE
C00C	10 FC	DJNZ C00AH
C00E	76	HALT

DJNZ命令で繰り返えし処理をする最初の命令LD (HL), Aは、上から5行目にあります。マシン語では、77です。この命令が記憶されているメモリ番地は、C00AHです。したがって、アセンブリ言語では、下から2行目に示されているように、

DJNZ C00AH

と、直接2バイトのメモリ番地を書きますが、マシン語の場合は、このように、直接メモリ番地を書くわけにはいきません。

DJNZ命令のマシン語は、10 e - 2 です。

10 e - 2 の e - 2 のところには、1 バイトのメモリ番地で指定する相対アドレス指定です。つまり、e - 2 のメモリ番地から何バイト先のメモリ番地まで戻れ、という指定の仕方です。

この場合、e - 2 も数えます。したがって右のように、マシン語を逆に数えていきます。77 は e - 2 から数えて - 4 バイト先にあります。そこで、10 のあとにある e - 2 のところに、- 4 を書きます。もちろん - 4 は10進数ですから、そのまま書くわけにはいきません。これを16進数になおします。巻末に示した符号つき16進数表をみると、- 4 = FCH となりますから、

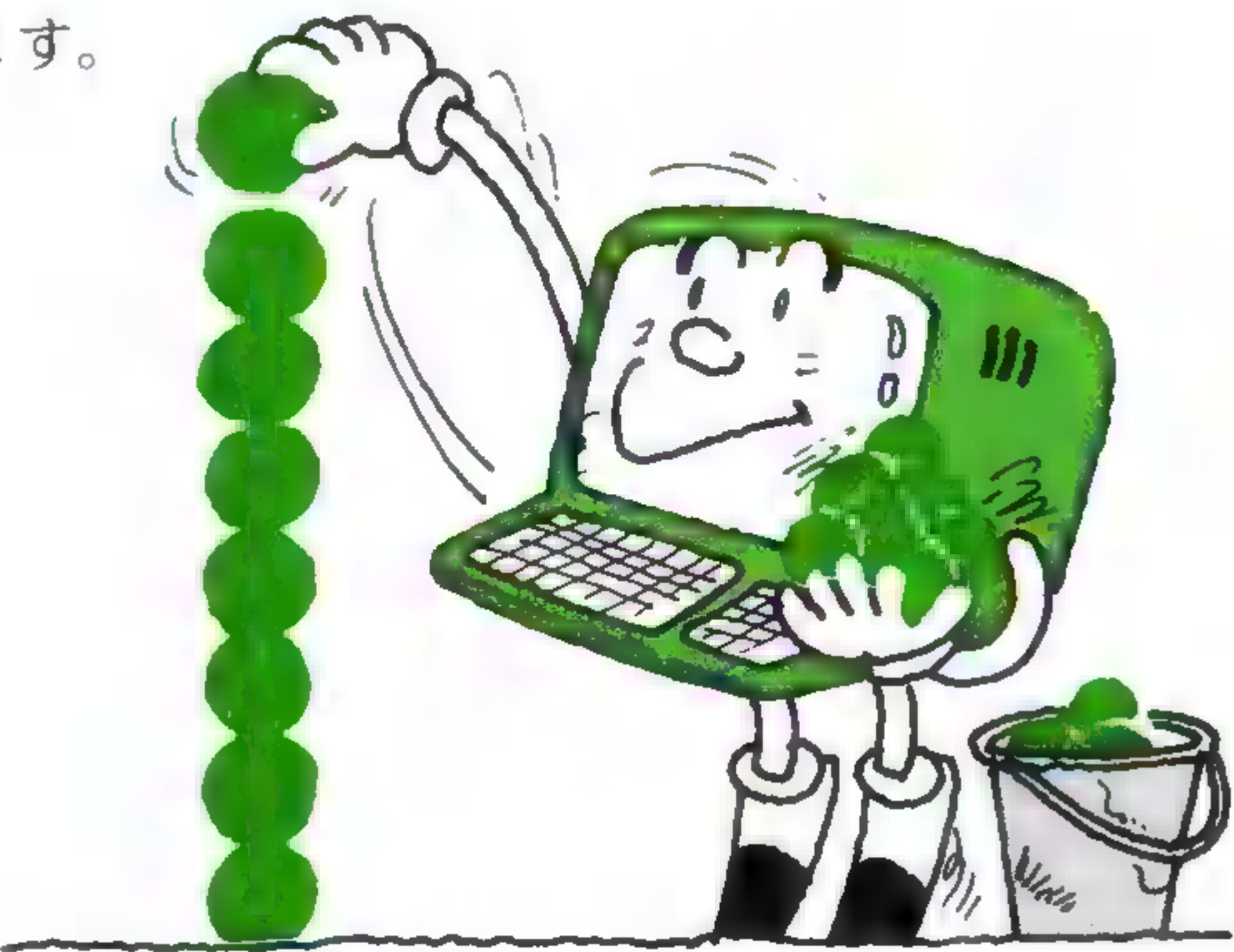
e - 2	- 1
10	- 2
19	- 3
77	- 4

10 FC ← e - 2 のところにFCと書く

となります。152頁に示したプログラムの下から2行目が、それです。このように数える理由は145頁で説明していますから参照してください。

このように、DJNZ命令で繰り返えし処理をさせると、Bレジスタにセットした値が0 になるまで、指定したメモリ番地に戻って繰り返えして処理を行います。この場合は、Bレジスタに10H（10進数で16）を代入しましたから16回繰り返えして、つぎの頁に示す実行結果のように、16個の●を画面の上から下に向けて表示していきます。

Bレジスタに代入した値が0 になるのは、DJNZ命令で指定したメモリ番地に戻るたびに、Bレジスタの値から1 を引いて戻るからです。Bレジスタの値が0 になると、DJNZ命令による繰り返えしをやめて、つぎのメモリ番地に記憶されている命令を実行します。



HALTはENDとトナリ

このプログラムでは、つぎのメモリ番地C00EHに記憶されているのはHALT命令です。この命令を実行すると、CPUが停止するため、そのまま動かなくなりますから、リセットボタンを押して、解除してください。

この場合、プログラムの最後にHALT命令を使ったのは、JP命令でメモリ番地5C66Hにジャンプさせて、ふたたびマシン語のコマンドを受けつけさせるようにすると、表示される*印で、画面の●が消えてしまうからです。

これで、●を画面の上から下に向けて表示していく、マシン語のプログラムができました。このプログラムを入力して実行させると、高速で処理されるので、●が画面の上から下に向かって描かれていくというよりも、●で一本の線が描かれるように見えますが、キャラクタを画面の上から下に向けて移動させるときは、このプログラムの方法で行います。

では、つぎにこのプログラムのダンプリストと、実行結果を示します。

```
C000 3E EC 21 28 F3 06 10 11
C008 78 00 77 19 10 FC 76 FF
```

ダンプリスト

```
mon
*GC000 ← 実行命令
```



上から下に表示

実行結果

auto so to list run

画面の下から上に向けて●を表示

16ビット（2バイト）の減算命令 SBC

148頁で、画面の上から下に向けて●を表示しました。ここではその逆に、画面の下から上に向けて●を表示していくことにします。

画面の上から下に向けて●を表示していくとき、16ビット（2バイト）の加算を必要としましたが、画面の下から上に向けて●を表示していくときは、16ビット（2バイト）の減算が必要になります。

16ビット（2バイト）の減算を行う命令は、**SBC命令**です。加算のときは、16ビット（2バイト）の場合も8ビット（1バイト）の場合も、おなじくADD命令でしたが、減算のときは、8ビット（1バイト）の場合、SUB命令を使います。

さて、16ビットの減算では、HL、BC、DEというレジスタペアを使います。8ビットの減算では、A、B、Cという8ビットのレジスタのままで、減算を行います。

つぎに、16ビットの減算命令SBCの形を示します。

SBC HL, XX

16ビットの減算命令SBCでは、うえに示したように、SBC命令のあとにHLレジスタを使います。そのあとにあるXXのところには、BC、DE、HLといったレジスタペアのうちの、いずれかひとつを選んで置きます。

そしてHLレジスタには、引かれる数を入れます。XXのところに置いたレジスタペアには、引く数を入れます。たとえば、つぎのような計算を行うとします。

FFFFH - 1111H =
↑ 引かれる数 ↑ 引く数

引かれる数FFFFHは、HLレジスタに入れます。引く数1111Hは、BCレジスタに入れることにします。

したがって、この引き算のプログラムは、つぎのようになります。

```
LD HL, FFFFH ← FFFFHをHLレジスタに代入
LD BC, 1111H ← 1111HをBCレジスタに代入
SBC HL, BC ← SBC命令で、HLレジスタからBCレジスタの値を引く
HALT
```

減算命令SBCを実行すると、HLレジスタのFFFFHからBCレジスタの1111Hを引きます。その結果のEEEEHは、HLレジスタに記憶されて残ります。これが、SBC命令を使った16ビットの減算です。

では、アセンブルです。

アセンブリ言語	マシン語
LD HL, FFFFH	21 FF FF
LD BC, 1111H	01 11 11
SBC HL, BC	ED 42
HALT	76

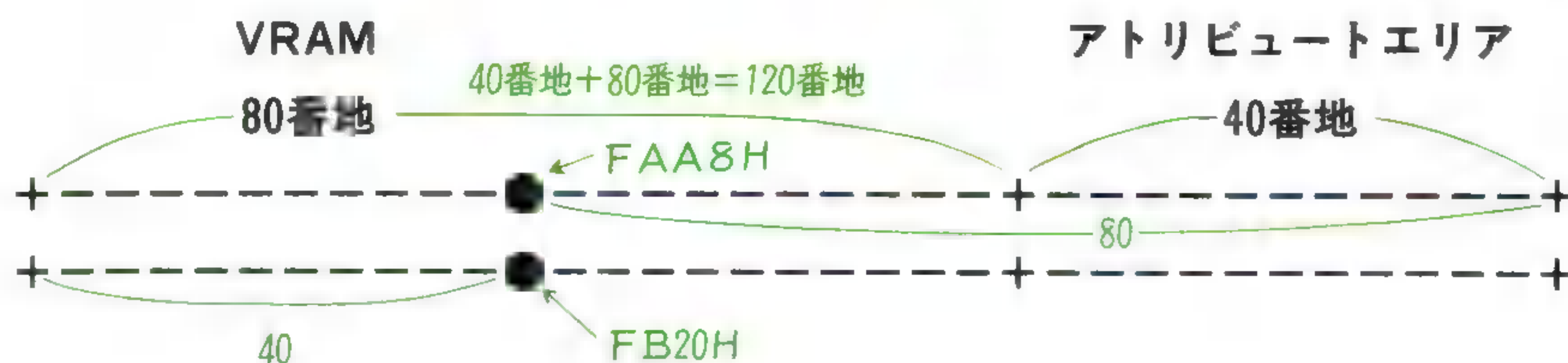
さて、16ビットの減算の仕方がわかったところで、●を画面の下から上に向けて表示することにします。

●を画面の下から上に内けて表示する

●を画面の下から上に向けて表示するとき、なぜ、16ビットの減算を使うのかというと、●を画面の上から下に向けて表示する場合と逆だからです。つまり、巻末に示したVRAMのメモリ番地表を見るとわかるように、●を表示させるVRAMのメモリ番地が、画面の上に行くにしたがって、小さくなっていくからです。また、VRAMのメモリ番地がFB20Hというように16ビット（2バイト）だからです。

たとえば、最初の●を画面の下のメモリ番地FB20Hに表示したとします。つぎにその真上の位置、つまりメモリ番地FAA8Hに●を表示するには、つぎの図に示すように、最初に●を表示したメモリ番地FB20Hから120（80番地+40番地=120）を引かないと、2個目の●を表示するメモリ番地FAA8Hを指定することができません。

画面の下から上に向けて●を表示



そこで、16ビットの減算が必要になるのです。

では、プログラムを作ります。まず、LD命令で、画面に表示する●のキャラクタコードECHを、Aレジスタに代入します。

LD A, ECH ← ●のキャラクタコードECHをAレジスタに代入

この場合は、●を画面の下から上に向けて表示していきますから、1個目の●を表示するVRAMのメモリ番地FB20Hを、LD命令でHLレジスタに代入します。

LD HL, FB20H ← メモリ番地FB20HをHLレジスタに代入

139頁で、ベーシックのFOR～NEXT命令とおなじように、繰り返えし処理するDJNZ命令について説明しました。このDJNZ命令は、Z-80CPUに追加されたZ-80CPU特有の命令で、非常に便利な命令です。このDJNZ命令を使うと、プログラムが非常に短くなりますから、ここでも、このDJNZ命令を使います。

140頁で説明したように、DJNZ命令で繰り返えし処理をする場合、繰り返えす回数は、Bレジスタにセットします。

つまり、画面の下から上に向けて17個の●を表示させるとしたら、Bレジスタに17をセットするわけです。

17は10進数ですから、そのままは使うことができません。17を16進数になおすと、11Hです。この11HをLD命令を使って、Bレジスタにセット、つまり代入します。

LD B, 11H ← 繰り返えす回数17をBレジスタに代入

繰り返えす回数をBレジスタに代入したら、繰り返えし処理する命令を並べていきます。この場合は、まず●を表示する命令です。●を画面に表示するには、VRAMのメモリ番地を指定して、そこにAレジスタに代入したキャラクタコードECHを、LD命令で代入してやればよいのです。この画面表示の命令は、つぎのようになります。

LD (HL), A

SBC命令で、HLレジスタから120を引く

137頁で、複数個のキャラクタを画面に表示していくとき、まず、DEC命令を2回実行させて、HLレジスタに記憶されている内容から2を引きました。そして、その値を、

LD (HL), A

のカッコのなかのHLに送って、カッコのなかのHLの値を変化させて、キャラクタを表示しました。このように、HLレジスタから1とか2とかの小さい数を引くようなときには、DEC命令を使ったほうがよいのです。

この場合は、さきに説明したように、HLレジスタの内容から120という大きな数を引く関係から、SBC命令を使うわけです。この場合の減算は、

メモリ番地-120

です。この減算を行うには、さきに説明したように、それぞれの値を16ビットのレジスタに代入しなければなりません。

メモリ番地は引かれる数なので、HLレジスタに代入します。120は引く数なので、BC、DEというレジスタに代入します。

引かれる数であるメモリ番地は、すでに2行目でHLレジスタに代入していますから、新たにHLレジスタに代入する必要はないわけです。したがって、引く数である120をBCかDEレジスタに代入します。ここでは120をDEレジスタに代入することにします。もちろん、120は10進数ですから、そのままではDEレジスタに代入することはできません。120を16進数になおすと、

120 = 78H

となります。78Hは、8ビット（1バイト）の数ですから、このままではだめです。78Hに00をつけて、

78H = 0078H ← 78Hに00をつけて16ビットにする

16ビットの数にして、DEレジスタに代入します。

LD DE, 0078H ← 120をDEレジスタに代入

メモリ番地から引く数120をDEレジスタに代入したら、HLレジスタのメモリ番地からDEレジスタの120を、SBC命令で引きます。

SBC HL, DE ← SBC命令で引く

SBC命令が実行されて減算された結果の値は、HLレジスタに記憶されます。つまり、2行目でHLレジスタに代入したメモリ番地FB20Hから、DEレ

ジスタに代入した0078Hを引いたとすると、

FB20H-0078H=FAA8H

減算された結果のFAA8Hは、HLレジスタに記憶されるというわけです。

ここで、参考のために、巻末に示したVRAMのメモリ番地表をみてください。
SBC命令で算出されたFAA8Hが、2行目でHLに代入したメモリ番地FB20Hの真上にあることがわかんと思います。

さて、SBC命令で算出された値を、LD (HL), AのカッコのなかのHLに送って、その位置に●を表示させます。

このことを、Bレジスタにセットした17回繰り返えさせます。そうすることで●を画面の下から上に向けて表示させていくことができます。ところで、このことを17回繰り返えさせるためには、DJNZ命令で、繰り返えし処理する最初の命令LD (HL), Aが記憶されているメモリ番地に戻します。

DJNZ命令で戻るメモリ番地の指定

DJNZ命令で繰り返えし処理をするには、繰り返えし処理をする最初の命令が記憶されている、メモリ番地に戻さなければなりません。この場合は、LD (HL), Aが記憶されているメモリ番地です。

ただし、DJNZ命令の場合、繰り返えし処理のために戻るメモリ番地の指定は、2バイトのメモリ番地を直接指定する絶対アドレス指定ではなく、1バイトで指定する相対アドレス指定です（145頁参照）。

したがって、プログラムをメモリ上に割り当てておかないと、DJNZ命令で戻るメモリ番地の指定ができませんから、これまで説明してきたプログラムをD000番地から割り当てることになります。

メモリ番地	マシン語	アセンブリ言語
D000	3E EC	LD A, ECH
D002	21 20 FB	LD HL, FB20H
D005	06 11	LD B, 11H
D007	11 78 00	LD DE, 0078H
D00A	77	LD (HL), A
D00B	ED 52	SBC HL, DE
D00D	10 FB	DJNZ D00AH
D00F	76	HALT

DJNZ命令で、繰り返し処理をする最初の命令LD (HL), Aは、上から5行目にあります。マシン語になおすと、77です。この命令を記憶させるメモリ番地はD00AHです。したがって、アセンブリ言語には、下から2行目に示されているように、

DJNZ D00AH

と、直接2バイトのメモリ番地を書きますが、マシン語のほうには、このように、直接2バイトのメモリ番地を書くわけにはいきません。

DJNZ命令のマシン語は、10 e-2です。

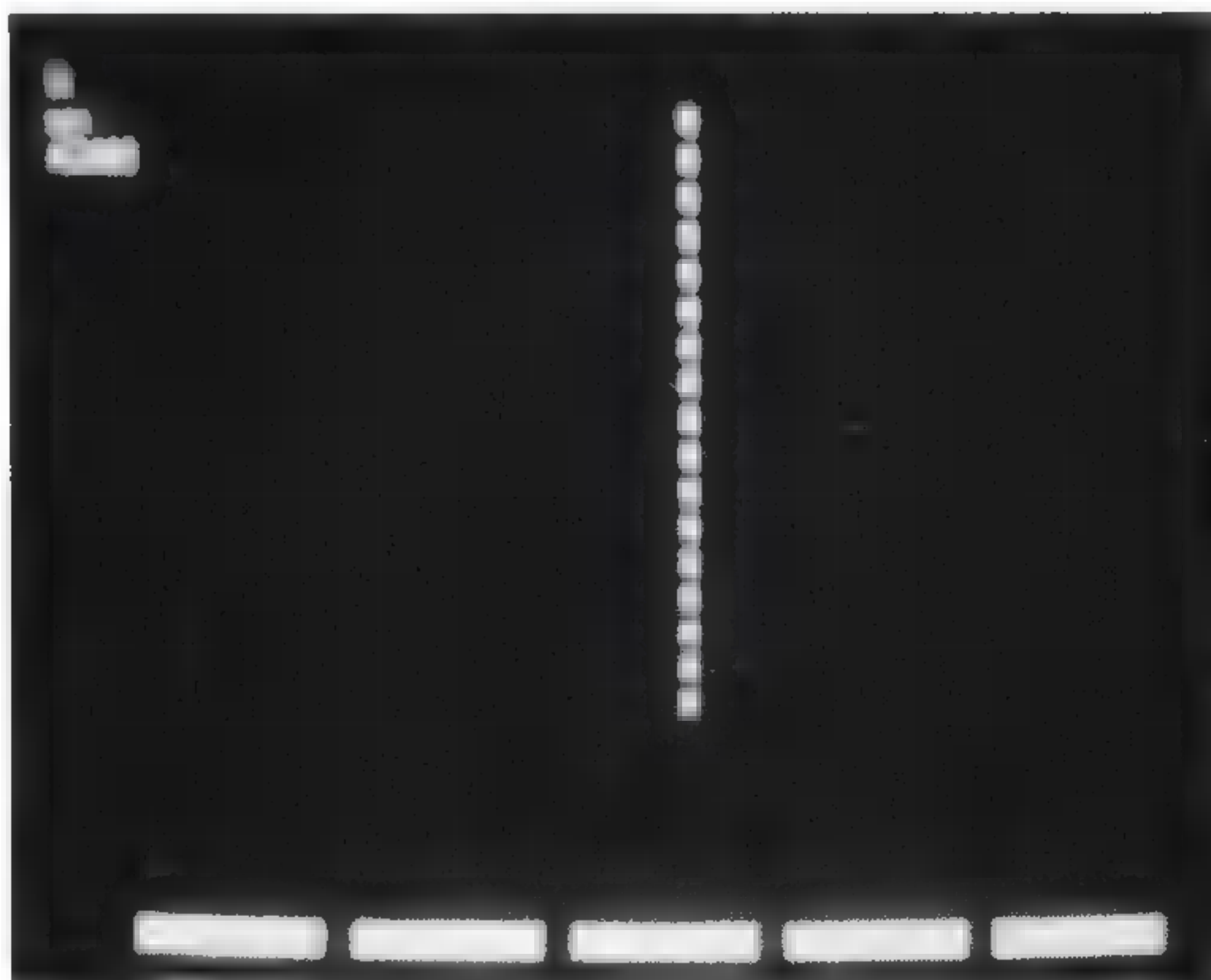
10 e-2のe-2のところは、1バイトのメモリ番地で指定する相対アドレス指定です。つまり、e-2のメモリ番地から、何バイト先のメモリ番地まで戻れという指定の仕方です。

この場合、e-2も数えます。したがって右のように、マシン語を逆に数えていきます。77はe-2から数えて-5バイト先にあります。そこで、10のあとにあるe-2のところに、-5を書きます。もちろん、-5は10進数ですから、そのまま書くわけにはいきません。これを16進数になおします。巻末に示した符号つき16進数表をみると、-5=FBHとなりますから、

e-2	-1
10	-2
52	-3
ED	-4
77	-5

10 FB ← e-2のところにFBHを■く

となります。159頁のプログラムの下から2行目が、それです。なぜ、このように数えて指定する理由は145頁で説明していますから参照してください。



画面の下から上に向けて●を表示

DJNZ 命令は、Bレジスタに代入した値だけ指定したメモリ番地にジャンプして、繰り返えし処理を行います。この場合は、11H（10進数で17）を代入しましたから、17回繰り返えして、つぎの実行結果に示すように●を画面の下から上に向けて表示していきます。

Bレジスタに代入した値が0になると、DJNZ 命令による繰り返えしは中止されて、つぎのメモリ番地に記憶されている命令を実行します。この場合はD00FH番地のHALT命令です。HALT命令は、ENDとおなじです。HALT命令については154頁を参照してください。

さてこれで●を下から上に向けて画面に表示していくマシン語のプログラムができました。実行させてみるとわかりますが、高速で処理されるので、ただ●で線が描かれるように見えますが、キャラクタを下から上に向けて移動させるときは、このプログラムの方法で行います。

ではつぎに、このプログラムのダンプリストと、実行結果を示します。

```
D000 3E EC 21 20 FB 06 11 11
D008 78 00 77 ED 52 10 FB 76
```

ダンプリスト

```
mon
*60000 ← 実行命令
```

下から上
に●を
表示

実行結果

4 auto auto list run

画面設定の仕方

CALL命令でモニタサブルーチンと呼ぶ

ベーシックでキャラクタを表示するときは、WIDTH命令で画面設定を行います。

WIDTH 80, 25 横80桁、縦25行の画面設定

WIDTH 40, 25 横40桁、縦25行の画面設定

マシン語では、ベーシックのWIDTH命令のような便利な命令はありません。したがって画面設定を行うときは、つぎに示すようにダイレクトモードで、

WIDTH 80, 25
ダイレクトモードで入力してもよい

WIDTH 40, 25

と入力して、**RET** キーを叩いて画面設定をするか、またCALL命令を使って、モニタサブルーチンを呼び出して、画面設定を行います。

CALL命令は、つぎのような形をしています。

CALL <メモリ番地>

このCALL命令は、マシン語でサブルーチンのあるプログラムを作ったときも、メインルーチンからサブルーチンを呼び出すために使います。ベーシックのGOSUB～RETURN命令のGOSUB命令です。

画面設定を行う場合、CALL命令のあとのメモリ番地には、093AHを書きます。このメモリ番地に、画面設定を行うモニタサブルーチンが記憶されています。メモリ番地093AHは、ROMにふられているメモリ番地であることはおわかりでしょう。つまりCALL命令で、パソコンに内蔵されているマシン語を利用するわけです。パソコンに内蔵されているマシン語を利用すると、非常に便利なこともありますから、パソコンに内蔵されているマシン語の利用方法をおぼえると役に立ちます。

さて、つぎに示すように、

CALL 093AH

CALL命令のあとに、メモリ番地093AHを書いただけでは、画面設定は行われません。

BレジスタとCレジスタに値をセット

まず、最初にBレジスタとCレジスタに値をセットします。Bレジスタには桁数をセットします。Cレジスタには行数をセットします。

Bレジスタ=桁数

Cレジスタ=行数

では、横80桁、縦25行の画面設定を行ってみることにします。

10進数の80は、16進数では50H、10進数の25は、16進数では19Hです。したがって、この値をLD命令でBレジスタとCレジスタに代入します。

LD B, 50H 10進数の80を入力

LD C, 19H 10進数の25を入力

そして、CALL命令で、画面設定のモニタサブルーチンのメモリ番地093AHを呼び出します。

CALL 093AH

これで画面は横80桁、縦25行に設定できますが、これだけで実行させると画面は横80桁、縦25行表示に設定されますが、エラーになります。マシン語のプログラムの最後には、HALTか、それに類する命令を置かなければなりません。ひとまずJP命令で、マシン語のコマンドを受けつけるモニタに、ジャンプさせることにします。

JP 5C66H

では、アセンブルです。

アセンブリ言語	マシン語
LD B, 50H	06 50
LD C, 19H	0E 19
CALL 093AH	CD 3A 09
JP 5C66H	C3 66 5C

なにかのプログラムのまえに、画面設定のプログラムを置くときは、最後にあるJP 5C66は4行目から取って、プログラムの最後につけます。

では159頁に示してある●を画面の下から上に向けて表示していくプログラム

つぎに、このプログラムのダンプリストを示します。

ダンプリスト

つぎに実行結果を示します。

実行結果

CALL 093AHで画面設定を80×25
モードにしての実行結果

なお、メモリ番地093AHのモニタサブルーチンは、PC-8001、PC-8001MKII、PC-8801・MKII（N-BASICモード）・MKIISR（N-BASICモード）のものです。ほかの機種の場合は、モニタサブルーチンのメモリ番地を確認して、そのメモリ番地をCALLしてください。

画面の下から上に向けて キャラクタを移動する

まえに表示したキャラクタを消すには **XOR A**

さて109頁から164頁までで、いろいろなマシン語の命令を使ってキャラクタを画面に表示しました。ここでは、キャラクタの移動を取りあげて説明することにします。画面の下から上へのキャラクタの移動です。

ベーシックで、キャラクタを移動させる方法は知っていると思います。

キャラクタを表示する

キャラクタが表示されている位置に空白を表示して消す

これの繰り返しです。マシン語の場合もおなじです。

画面の下から上へキャラクタを表示する方法は155頁で説明した方法です。この方法に、表示したキャラクタを消す命令が追加されるだけです。

追加する命令は、**XOR命令**といって、つぎのような形をしています。

XOR X

このXORは、排他的論理和を計算する命令です。XORのあとにあるXのところには、つぎに示すように、AからLまでのレジスタを置きます。

X=A、B、C、D、E、H、L

計算するという場合、あるレジスタの内容と、あるレジスタの内容との間で行うものと考えるのがふつうですが、XORのあとには、レジスタを置くXがひとつしかありません。実はXOR命令は、

Aレジスタの内容

Xに置かれたレジスタの内容

との間で計算が行われると決まっているからです。たとえば、

XOR B

とXにBレジスタを置くと、Aレジスタの内容とBレジスタの内容との間で計算が行われます。そして、計算された結果は、Aレジスタに記憶されて残ります。

では、どんな計算を行うのでしょうか。まず、真理値表を示します。

0	0	0	0と0のときは、0になる
0	1	1	0と1のときは、1になる
1	0	1	1と0のときは、1になる
1	1	0	1と1のときは、0になる

さて、Aレジスタの内容が25H、Bレジスタの内容が43Hとします。するとXOR命令は、つぎのような計算を行います。

25H ✕ 43H

✕の記号は、排他的論理和を表わす記号です。もちろん、25Hと43Hは16進数ですから、計算はつぎのように2進数に変換されて行われます。

25H → 0010 0101

43H → 0100 0011

一番右端は、両方とも1ですから、

1 ✕ 1 = 0

となります。真理値表をみてください。そうになっています。つぎは、0と1ですから、

0 ✕ 1 = 1

となります。この計算を、ビットごとに順に左へ行っていくと、

25H → 0010 0101

43H → 0100 0011

.....

66H → 0110 0110

XOR

ケイの下に、01100110という2進数が求められます。この2進数を16進数に変換すると66Hとなります。

これが、排他的論理和という計算の仕方です。このようなことが、何に役立つのかと考えるかも知れません。ここで大切なことは、0と0が0、1と1が0になるということです。

XOR Aで、空白のキャラクタコードを作る

XOR ×の×のところに、さきに示したように、**A**レジスタをも置くことができます。

XOR A

すると、どうなるでしょうか。

Aレジスタの内容

Aレジスタの内容

との間で、排他的論理和の計算が行われることになります。

たとえば、**A**レジスタに**ECH**を代入したとします。**ECH**は、●のキャラクタコードです。

LD A, ECH ● **ECH**を**A**レジスタに代入

XOR A ● 排他的論理和の計算を行う

すると、さきのような計算が行われます。

ECH	1 1 1 0	1 1 0 0
	XOR	
ECH	1 1 1 0	1 1 0 0
<hr/>		
	0 0 0 0	0 0 0 0

ケイの下に、00000000という2進数が求められました。この2進数を16進数に変換すると、00Hです。この00Hは**A**レジスタに記憶されて残ります。ベーシックでプログラムを組んだことがある人なら、この00Hが何であるかわかるとおもいますが、この00Hは、空白のキャラクタコードなのです。したがって、

LD (HL), A ● キャラクタを表示

XOR A ● 空白のキャラクタコードを算出して**A**レジスタに記憶

LD (HL), A ● 空白を表示

とすれば、キャラクタが画面に表示され、つぎに空白が表示されて、表示したキャラクタを消す、ということが行われるのです。

このことを、繰り返えしの命令**DJNZ**を使って繰り返えすと、キャラクタの移動が行われるというわけです。

●を画面の下から上へ移動する

では、●を画面の下から上に向けて移動するプログラムを作ります。

ここでは、画面設定を変更して、横40桁、縦25行表示にします。画面の下のファンクションキーの内容表示も消します。

画面設定の変更はダイレクトモードでつぎのように入力して行います。

WIDTH 40, 25

CONSOLE ,, 0

DJNZ命令で戻すところはLD A, ECH

●を移動するときに注意することは、DJNZ命令で繰り返えし処理をする最初の命令を、

LD A, ECH ●のキャラクタコードを代入

にすることです。なぜなら、表示した●を消すために、XOR命令を使って、Aレジスタの内容を00Hにしてしまうため、繰り返えしLD命令でAレジスタにECHを代入しないと、●が表示されていかない、ということになるからです。

このように、DJNZ命令で繰り返えす最初の命令をLD A, ECHにすることから、このプログラムでは、まず、●を表示するVRAMのメモリ番地をHLレジスタに代入することからはじめます。

この場合、ダイレクトモードで、画面設定を変更しますから、画面の一番下のVRAMのメモリ番地を使うことができます。画面の真中で、画面の一番下のメモリ番地はFE68Hですから、このメモリ番地を、LD命令でHLレジスタに代入します。


LD HL, FE68H ●を表示するメモリ番地を代入


つぎは、DJNZ命令で、繰り返えす回数をBレジスタにセットします。画面の一番下から画面の一番上までは25行ですから、25回繰り返えせばよいということになります。10進数の25は、16進数で19Hです。そこで、LD命令でBレジスタに19Hを代入します。

LD B, 19H DJNZ命令で繰り返えす回数を代入



このあとに、DJNZ命令で繰り返えす命令を並べていきます。まず、表示する●のキャラクタコードECHを、LD命令でAレジスタに代入します。

LD A, ECH ← のキャラクタコードECHを代入


つぎは、の表示です。ECHはAレジスタに代入していますから、それをVRAMのメモリ番地を指定するHLレジスタに、LD命令で代入します。

LD (HL), A ← を表示する



XOR Aで、00Hを算出する

さて、AレジスタのECHを、HLレジスタに代入してを表示したら、つぎは、表示したを消すために00Hを算出します。00Hの算出は、さきに説明したように、



XOR A ← 空白のキャラクタコード00Hを算出


です。XOR命令で算出した00Hは、Aレジスタに記憶されています。したがって、表示したを消すために、Aレジスタの00Hを、LD命令でHLレジスタに代入して表示します。


LD (HL), A ← 空白を表示

この場合のHLレジスタの内容、つまりメモリ番地は変化していませんから、を表示したメモリ番地とおなじメモリ番地に00Hが表示されて、表示されているが消されます。

SBC命令で、つぎに表示するメモリ番地を算出

Aレジスタの00Hを表示して、表示しているを消したら、つぎにを表示するメモリ番地を計算して出します。メモリ番地は16ビット（2バイト）です。したがって、計算は16ビットのレジスタで行います。つまり、レジスタペアです。このメモリ番地の算出については156頁を参照してください。

さて、つぎにを表示するためのメモリ番地を計算するために、まず、LD命令で、120をDEレジスタに代入します。120は10進数です。16進数では78Hですが、8ビットの数と16ビットの数の計算はできませんから、78Hに00Hをつけて、0078Hという16ビットの数にします。したがって、0078Hを、LD命令でDEレジスタに代入することになります。

LD DE, 0078H ← 16ビットの数にして代入する

0078HをDEレジスタに代入したら、HLレジスタの内容（メモリ番地）からDEレジスタの内容（0078H・10進数で120）を、減算の命令SBCを使って引きます。

DJNZ命令で、繰り返し処理するために戻す

そして、これまで説明した命令を繰り返し処理するために、DJNZ命令を使います。

10 e-2 ← DJNZ命令のマシン語

DJNZ命令のあとにあるe-2には、相対アドレスを指定しなければなりません。したがって、これまで説明した命令をまとめて、メモリ番地を割り当てることにします。メモリ番地D000Hからです。

メモリ番地	マシン語	アセンブリ言語
D000	21 68 FE	LD HL, FE68H
D003	06 19	LD B, 19H
D005	11 78 00	LD DE, 0078H
D008	3E EC	LD A, ECH
D00A	77	LD (HL), A
D00B	AF	XOR A
D00C	77	LD (HL), A
D00D	ED 52	SBC HL, DE
D00F	10 F7	DJNZ D008H
D011	C3 66 5C	JP 5C66H

さて、DJNZ命令で、繰り返し処理をする最初の命令LD A, ECHは、メモリ番地D008Hに記憶されています。したがって、うえに示したプログラムの下から2行目に見られるように、アセンブリ言語のほうには、そのメモリ番地D008Hをそのまま書きますが、マシン語のほうには、そのメモリ番地をそのまま書くことができません。なぜなら、1バイトで指定する相対アドレス指定だからです。このことについては144頁で説明しているので、その頁を参照してください。

ここでは、10 e-2のe-2のところに、

10 F7 ← 相対アドレス

F7とあります。なぜF7となっているかというと、つぎのようにe-2のところから逆に、-1、-2と数えていくからです。

e - 2	- 1	7 7	- 7
1 0	- 2	E C	- 8
5 2	- 3	3 E	- 9
E D	- 4		
7 7	- 5		
A F	- 6		

このように、e - 2 のところから逆に数えていくと、LD A, ECH命令の最初のマシン語3 Eが、- 9 個目にあることがわかります。その- 9 をDJNZ命令のe - 2 のところに書くわけです。もちろん、- 9 は10進数ですから、巻末にある符号つき16進数表をみて、16進数になおして書きます。

- 9 = F 7 H ← - 9 は F 7 H になる

DJNZ命令のマシン語にF 7 Hを指定すると、Bレジスタにセットした回数だけ、F 7 Hのところに戻って繰り返し処理を行います。

DJNZ命令は、指定されたメモリ番地に戻るたびに、Bレジスタの内容から1を引いていきます。そして、Bレジスタの内容が0になると、繰り返しをやめて、つぎのメモリ番地に記憶されている命令の実行に移ります。

このプログラムの場合はJP 5 C 6 6 Hですから、JP命令でメモリ番地5 C 6 6 Hにジャンプして、ふたたびマシン語のコマンドを受けつける*印を表示します。

これで、画面の下から上に向かって●を移動させるマシン語のプログラムができあがりました。残念なことに、このプログラムの実行結果を示すことができません。

なぜなら、XOR A命令で算出される0 0 Hで、画面に表示された●が消されていくからです。それが高速に行われるので、画面では●の移動を確認することができません。もし、確認したいとするなら、ベーシックの命令で画面いっぱいに■を表示しておいて、そのあと、プログラムを実行させると、●が通ったところが抜けるので、●が下から上に移動していったことが確認できます。

なお178頁で、命令の実行を遅らせるウェイト・ループについて説明しています。このウェイト・ループを使うと、それぞれの命令の実行を遅らせることができるので、●の移動を確認することができるようになります。

画面の上から下に向けて キャラクタを移動する

ポイント は XOR A と、ADD HL, X

165頁で、画面の下から上に向けて、キャラクタを移動していく方法について説明しました。キャラクタを画面の下から上に向けて移動していく場合は、

```
LD  (HL), A  ← 画面にキャラクタを表示
XOR  A  ← 空白を算出
LD  (HL), A  ← 空白を表示して、キャラクタを消す
SBC  HL, DE  ← 引き算で、つぎに表示するメモリ番地を算出
```

これらの命令を、DJNZ命令で、必要な回数だけ繰り返えさせました。

画面の上から下に向けてキャラクタを移動していく方法も、画面の下から上に向けてキャラクタを移動していく方法と、さして変わるところはありません。変わるところといえば、16ビットの引き算の命令SBCが、16ビットの足し算の命令ADDに変わるということだけです。まず、そのことを知ってもらうために、両方のアセンブリ言語のプログラムを示します。

画面の下から上への移動	画面の上から下への移動
LD HL, FE68H	LD HL, メモリ番地
LD B, 19H	LD B, 繰り返えす回数
LD DE, 0078H	LD DE, 0078H
LD A, ECH	LD A, キャラクタコード
LD (HL), A	LD (HL), A
XOR A	XOR A
LD (HL), A	LD (HL), A
SBC HL, DE	ADD HL, DE
DJNZ D008H	DJNZ D008H
JP 5C66H	JP 5C66H

画面の上から下にキャラクタを移動するプログラムの 1行目、2行目、4行目

画面の上から下に向けてキャラクタを移動する

では、それぞれのレジスタに代入する値が決まっていないので、何が入るのかは文字で示しましたが、使う命令は、左側に示したプログラムとおなじです。

このふたつのプログラムを比べて見ると、違う点は、さきに説明したSBC命令と、ADD命令のひとつだけということがわかんと思います。

もうひとつ注意しておいてもらいたいことは、DJNZ命令のあとのメモリ番地が、両方ともおなじD008Hとなっています。アセンブリ言語で書くときは、DJNZ命令で戻るメモリ番地をそのまま書くので、おなじメモリ番地になりますが、アセンブルするときは、相対アドレスで指定するので、おなじにはならないということです。

では、画面の上から下へキャラクタを移動する方法について説明します。

画面設定をする

画面の上から下へのキャラクタの移動なので、画面設定を縦25行表示に変更することにします。横の桁数は40のままです。ファンクションキーの表示もそのままにしておきます。この画面設定をベーシックで書くと、

WIDTH 40, 25

となりますが、これをアセンブリ言語で書くと、つぎのようになります。

LD B, 28H ・————— Bレジスタに、横の桁数40を代入

LD C, 19H ・————— Cレジスタに、縦の行数25を代入

CALL 093AH ・————— 画面設定のモニタサブルーチンを呼び出す

画面設定の説明は162頁で行っていますので、その頁を参照してください。

VRAMのメモリ番地などを決める

つぎは、巻末に示したVRAMのメモリ番地表をみて、キャラクタを最初に表示するメモリ番地を決めます。この場合は、画面の真中で、画面の一番上のメモリ番地F328Hを選ぶことにします。選び出したF328Hを、LD命令でHLレジスタに代入します。

LD HL, F328H ・————— メモリ番地を代入

画面の上から下へキャラクタを移動するようなときは、繰り返えしの命令DJNZを使って、キャラクタを表示したり、キャラクタを消したりする命令などを繰り返えし処理させます。このDJNZ命令を使うと、プログラムが非常に短くなるからです。

DJNZ 命令を使うときは、繰り返し処理する回数を B レジスタに代入することになっています。繰り返し回数は、キャラクタをどこまで移動させるかで決めます。画面の下には、ファンクションキーの内容が表示されていますから、画面の上から 21 行目まで移動させることにしましょう。10 進数の 21 は、16 進数で 15 H ですから、LD 命令で B レジスタに 15 H を代入します。

```
LD B, 15H ← 繰り返し回数を B レジスタに代入
```

画面設定のところで、B レジスタに 28 H を代入しました。ここで、B レジスタに 15 H を代入しています。LD B, 15 H が実行されると、B レジスタの内容は新たに代入した値、つまり 15 H になります。レジスタも、ベシックの変数とおなじ働きです。このことを、おぼえておいてください。

キャラクタの表示と消去

さてつぎは、表示するキャラクタは ▼ とします。▼ のキャラクタコードは 27 H ですから、27 H を LD 命令で A レジスタに代入します。

```
LD A, 27H ← ▼ のキャラクタコード 27 H を代入
```

A レジスタに 27 H を代入したら、A レジスタの内容 27 H の表示です。表示は A レジスタの内容を LD 命令で、VRAM のメモリ番地を指定する HL レジスタに代入すればよいのですから、つぎのようになります。

```
LD (HL), A ← ▼ を表示
```

キャラクタを表示したら、そのキャラクタを消します。キャラクタを消すには、空白のキャラクタコードが必要です。空白のキャラクタコードを作り出すのが、XOR A 命令です。ここで大切なことは、XOR のあとに A を置くということです。XOR A とすると、A レジスタの内容同士で排他的論理和を求める計算を行います。このことについては、167 頁を参照してください。

A レジスタに代入したキャラクタコードは 27 H でした。したがって、

27H	0010	0111
	XOR	
27H	0010	0111

	0000	0000

画面の上から下に向けてキャラクタを移動する

となって、2進数の00000000が求められます。この結果は、Aレジスタに記憶されます。2進数の00000000を16進数になおすと、00Hです。したがって、Aレジスタに空白のキャラクタコード00Hを代入したことで、おなじことになるのです。

このように、XOR A命令によって00Hが算出されて、Aレジスタは00Hになっていますから、表示されているキャラクタを消すために、Aレジスタの内容を、LD命令でHLレジスタに代入して表示します。

LD (HL), A ← 空白を表示

キャラクタを表示するメモリ番地の計算

最初のキャラクタは、VRAMのメモリ番地F328Hに表示しました。上から下にキャラクタを移動していくには、つぎのキャラクタはメモリ番地F328Hの真下にある、メモリ番地F3A0Hに表示しなければなりません。

そこでHLレジスタの内容（メモリ番地）に120を足します。なぜ120を足すのかについては149頁を参照してください。

120は10進数ですから、16進数になおします。16進数では78Hです。

メモリ番地はF328Hというように、16ビット（2バイト）の数です。78Hは、8ビット（1バイト）の数です。16ビットと8ビットの数とでは計算できませんから、78Hのまえに00Hをつけて、0078Hという16ビットの数にします。そして、HLレジスタの内容（メモリ番地）に0078Hを足すために、LD命令で16ビットのレジスタに代入します。ここでは、DEレジスタに代入することにします。

LD DE, 0078H ← 足す数を代入

つぎにADD命令で、DEレジスタの0078Hを、HLレジスタの内容（メモリ番地）に足します。

ADD HL, DE ← HLレジスタの内容とDEレジスタの内容を足す

最初のメモリ番地はF328Hですから、つぎのように足し算されます。

F328H+0078H=F3A0H

足し算の結果のF3A0Hは、HLレジスタに記憶されて残ります。

ところでF3A0Hは、VRAMのメモリ番地表でみると、最初にキャラクタを表示したメモリ番地F328Hの真下にあるメモリ番地です。このようにHLレジスタの内容（メモリ番地）に0078H（10進数で120）を足してつぎに

表示するメモリ番地を指定していくわけです。

さて、キャラクタを表示し、表示したキャラクタを消す、ということを繰り返さないと、キャラクタの移動は行われません。そこで、DJNZ命令で、繰り返えし処理する最初の命令に戻します。

繰り返えし処理をする最初の命令は、LD A, 27Hです。なぜかという、XOR A命令で、Aレジスタの内容が00Hになってしまっている、新たに27Hを代入しないと、キャラクタが表示されないからです。

DJNZ命令で戻るメモリ番地の指定

さきに説明したように、DJNZ命令で繰り返えし処理をする最初の命令は、LD A, 27Hです。この命令に戻すには、1バイトの相対アドレスで指定しなければなりません。そこで、これまで説明してきた命令を、アセンブルして示します。

メモリ番地	マシン語	アセンブリ言語
D000	06 28	LD B, 28H
D002	0E 19	LD C, 19H
D004	CD 3A 09	CALL 093AH
D007	21 28 F3	LD HL, F328H
D00A	11 78 00	LD DE, 0078H
D00D	06 15	LD B, 15H
D00F	3E 27	LD A, 27H
D011	77	LD (HL), A
D012	AF	XOR A
D013	77	LD (HL), A
D014	19	ADD HL, DE
D015	10 F8	DJNZ D00FH
D017	C3 66 5C	JP 5C66H

DJNZ命令で、繰り返えし処理する命令LD A, 27Hは、メモリ番地D00FHに記憶されています。したがってアセンブリ言語のほうには、D00FHを書きます。DJNZ命令のマシン語10のほうには、F8と書かれています。これは、10進数の-8です。このようにDJNZ命令には、1バイトのメモリ番地で指定します。このような指定の仕方を、相対アドレス指定と呼んでいま

画面の上から下に向けてキャラクタを移動する

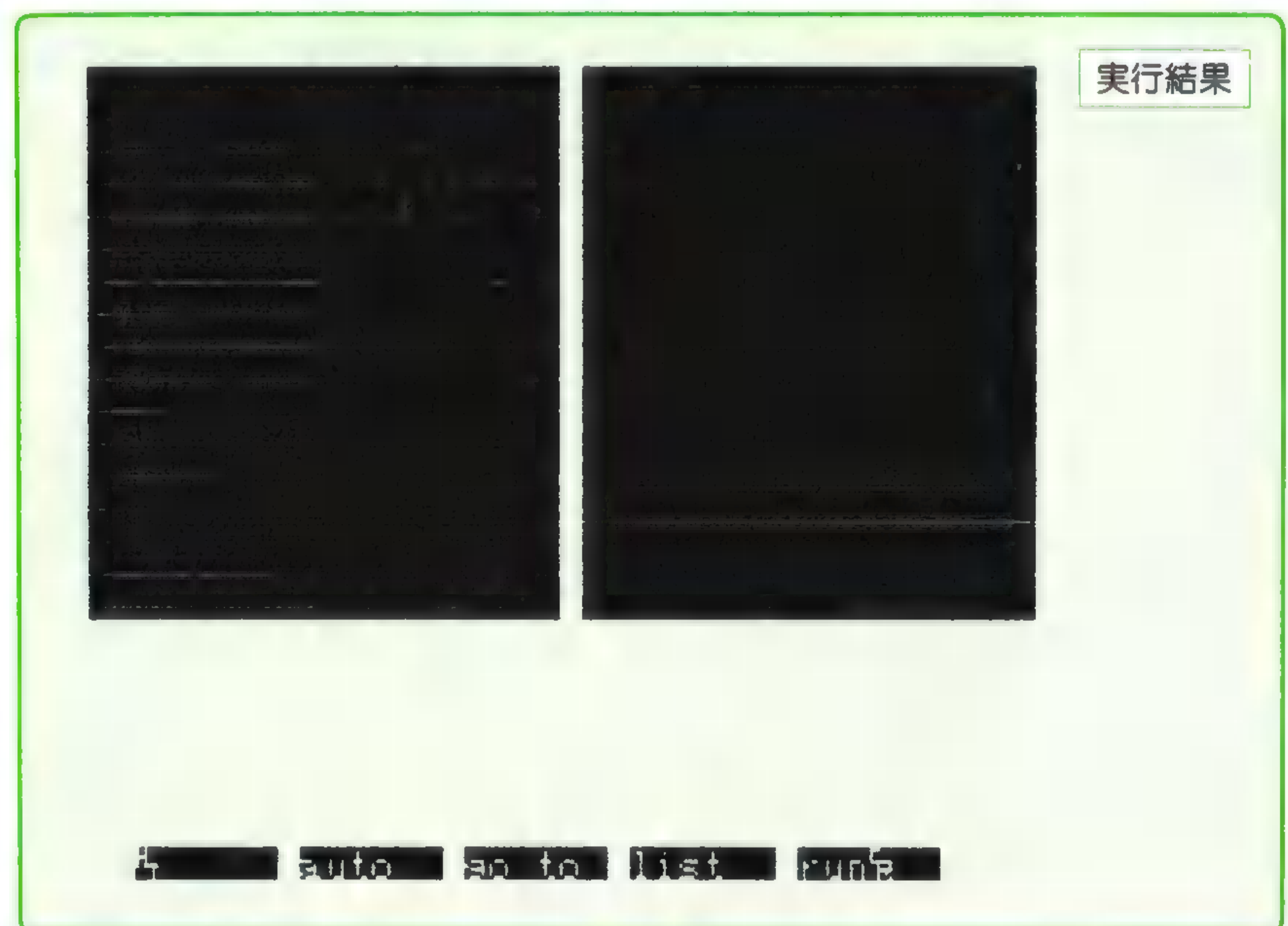
す。相対アドレス指定については145頁を参照してください。

DJNZ命令は、繰り返えし処理する最初の命令に戻るたびに、Bレジスタの内容から1を引きます。Bレジスタの内容が0になると、繰り返えし処理をやめて、つぎのメモリ番地の命令を実行します。この場合は、Bレジスタに15H（10進数で21）を代入しましたから、DJNZ命令で、キャラクタを表示したり、表示したキャラクタを消したりすることを21回繰り返えして、画面の上から下に向けて、キャラクタを移動させていきます。

Bレジスタの内容が0になると、つぎのメモリ番地のJP 5C66Hを実行します。そこで、JP命令でマシン語のコマンドを受けつけるメモリ番地にジャンプして、ふたたび*印を表示することになります。

さて、このプログラムを実行させてみても、非常に高速に処理されるため、キャラクタの移動を確認することができません。どうしても確認したいときはつぎの実行結果に示すように、何かを画面いっぱいに表示して実行させると、キャラクタの通ったところが抜けるので確認できます。

なお178頁で、命令の実行を遅らせるウェイト・ループについて説明します。このウェイト・ループを使うと、命令の実行を遅らせることができるようになるので、キャラクタの移動を確認することができます。



命令の実行を遅らせるには

ウェイト・ループ

165頁などでキャラクタを移動させましたが、キャラクタを表示する、キャラクタの表示を消す、という命令の実行が非常に高速に行われるため、画面には何も表示されませんでした。これではキャラクタの移動とはいえないので、

キャラクタを表示する

ウェイト・ループ

キャラクタの表示を消す

というように、ふたつの命令の間にウェイト・ループを入れて、キャラクタの表示を消す命令の実行を遅らせることにします。ベーシックのウェイト・ループは、FOR～NEXT命令ひとつですが、マシン語の場合、こんなに簡単ではありません。まず、ウェイト・ループの部分だけを説明します。

メモリ番地	アセンブリ言語
D009	LD B, FFH
D00B	LD A, FFH
D00D	DEC A
D00E	JP NZ D00DH
D011	DJNZ D00BH

アセンブリ言語の頭にふられているメモリ番地は、マシン語になおしたときのメモリ番地です。したがって、マシン語になおしてメモリ番地をふるべきなのですが、説明の都合上、このようにします。

うえに示した命令が、ウェイト・ループの働きをする命令です。このウェイト・ループの働きを追加すると、さらにほかの命令も必要となりますが、まずこの命令の働きを説明することにします。

さて、LD命令で、BレジスタにFFHを代入します。FFHは、10進数で255です。Bレジスタに代入したFFHは、D011番地にあるDJNZ命令の繰

り返えしの回数です。

```
LD B, FFH ← BレジスタにDJNZ命令の繰り返えしの回数を代入
```

つぎにLD命令で、Aレジスタにもおなじように、FFHを代入します。

Aレジスタに代入したFFHを、DEC命令を使って、0になるまで引かせていきます。DEC命令は、指定されたレジスタの内容から、1を引く命令です(132頁参照)。

```
LD A, FFH ← AレジスタにFFHを代入
```

```
DEC A ← Aレジスタの内容から1を引く
```

このふたつの命令だけでは、Aレジスタの内容が0になるまで引いていくことはできません。Aレジスタの内容から1ずつ引くことを、繰り返えさせる命令が必要になります。その命令が、

```
JP NZ DOODH
```

です。JP NZ命令は、

```
ゼロ・フラグが0のときメモリ番地にジャンプ
```

という働きをします。ベーシックでいうとすると、

```
IF NZ=0 THEN 行番号
```

ということになるかもしれません。ではゼロ・フラグとはなにかです。

フラグとは

さて、むずかしいフラグがでてきました。ここでは、必要なゼロ・フラグについてだけ説明します。

フラグは、CPUのなかにある8ビットレジスタで、メモリには第0ビットから第7ビットまでの8つの記憶場所があるように、フラグも8つの記憶場所を持っています。その8つの記憶場所は、つぎのように呼ばれています。

7	6	5	4	3	2	1	0	ビット ←番号
S	Z	X	H	X	PV	N	C	←記号
サイン	ゼロ	ハーフキャリー		パリティ		加/減算	キャリー	

さて、ゼロ・フラグは1ビットですから、0か1の値になります。ゼロ・フラグが1か0になるのは演算を行ったときで、

```
演算結果が0のとき = 1
```


演算結果が0でないとき=0

というように変化します。これがゼロ・フラグの働きです。

JP NZ命令は、ゼロ・フラグが0のときだけジャンプ

では、説明をもとに戻します。

LD A, FFH

DEC A

さきに説明したように、この2つの命令は、Aレジスタの内容（FFH）から1を引くという演算です。つまり、DEC Aを1回目実行すると、

FFH - 1 = FEH

となりますから、Aレジスタの内容は、まだ0ではありません。したがって、ゼロ・フラグは0になっています。

JP NZ命令は、ゼロ・フラグが0のとき、指定されているメモリ番地にジャンプするわけですから、この場合は、メモリ番地D00DHにジャンプすることになります。メモリ番地D00DHに記憶されている命令は、DEC A命令です。そこで、ふたたびAレジスタの内容から1を引きます。

FEH - 1 = FDH

つまり、Aレジスタの内容が0になるまで、

D00B LD A, FFH

D00D DEC A

D00E JP NZ D00DH

Aレジスタが0になるまで
繰り返す

このふたつの命令の間を繰り返すことになります。

DEC A命令によって1ずつ引かれていって、Aレジスタの内容が0になると、ゼロ・フラグは1になります。JP NZ命令は、ゼロ・フラグが0のときだけ、指定されたメモリ番地にジャンプするのですから、ゼロ・フラグが1になると、つぎのメモリ番地の命令の実行に移ります。

つぎのメモリ番地D011Hの命令は、DJNZ命令です。このDJNZ命令は、繰り返し処理の命令で、Bレジスタにセットされた値だけ、指定されたメモリ番地に戻って、繰り返し処理を行います。

D011 DJNZ D00BH

DJNZ命令に指定されているメモリ番地はD00BH、つまりLD A, FFHです。そこで、ふたたびLD命令でAレジスタにFFHを代入することになり

ます。ということは、またDEC A命令と、JP NZ命令を繰り返すわけです。このことを、Bレジスタで指定したFFH回、DJNZ命令で繰り返します。FFHは、10進数で255ですから、255回繰り返すことを、さらに255回繰り返すわけです。

このウェイト・ループをベーシックで書くと、つぎのようになります。

```
FOR I=255 TO 0 STEP -1
FOR J=255 TO 0 STEP -1
NEXT J
NEXT I
```

ベーシックで、こんなウェイト・ループを使ったらたまりません。マシン語でもかなり遅くなります。キャラクタの移動を速くしたいときは、

```
D009 LD B, FFH  FFHを64Hとかに変える
D00B LD A, FFH  FFHを64Hとかに変える
```

うえに示した命令のFFH（10進数で255）を、64H（10進数で100）とか数を小さくします。両方がおなじ数でなくてもかまいません。どちらか一方の数を小さくしても、キャラクタの移動は速くなります。

キャラクタを画面の下から上へ移動する

では、このウェイト・ループを165頁で説明した、キャラクタを画面の下から上に移動するプログラムに使ってみます。このウェイト・ループを使うと、ほかの新しい命令を使う必要がありますが、その命令については、その命令のところで説明することにします。キャラクタを画面の下から上に移動するプログラムは165頁で説明してありますし、ウェイト・ループの説明もすんでいるので、プログラムをさきに示します。アセンブリ言語にふられているメモリ番地は、マシン語にふるメモリ番地です。




```

D000    LD  HL, FC88H
D003    LD  DE, 0078H
D006    LD  B, 15H
D008    LD  A, ECH
D00A    LD  (HL), A
D00B    PUSH BC
D00C    LD  B, 64H
D00E    LD  A, 64H
D010    DEC  A
D011    JP  NZ  D010H
D014    DJNZ D00EH
D016    POP  BC
D017    XOR  A
D018    LD  (HL), A
D019    SBC  HL, DE
D01B    DJNZ D008H
D01D    JP  5C66H

```

ウェイトループ

それぞれの命令の動き

では、それぞれの命令の動きを説明していきます。

まず、最初にキャラクタを表示するVRAMのメモリ番地を、LD命令で、HLレジスタに代入します。

LD HL, FC88H

FC88Hは、画面の下、ファンクションキーの表示の上の位置で、画面の真中の位置です。ただし、縦25行表示の場合ですから、WIDTH命令で、

WIDTH 40, 25

画面設定を変更しなければなりません。

つぎに、LD命令で、キャラクタを移動させる繰り返えしの回数を、Bレジスタにセットします。この場合は、画面の下の方位置、メモリ番地FC88Hから画面の上まで、21行ありますから、21を16進数になおした15Hを、Bレジ

スタに代入します。

```
LD B, 15H
```

このBレジスタにセットした繰り返えす回数は、メモリ番地D01BHの繰り返えしの命令DJNZの繰り返えし処理する回数です。

```
D01B DJNZ D008H
```

このDJNZ命令は、メモリ番地D008Hから繰り返えし処理します。つまり、キャラクタを表示したり、表示したキャラクタを消したりなどを、21回行います。

さて、表示するキャラクタを、LD命令でAレジスタに代入します。

```
LD A, ECH
```

ECHは、●のキャラクタコードです。まえに取りあげたDJNZ命令は、このLD A, ECHに戻ってきます。ここに戻ってきて、繰り返えし表示するキャラクタをAレジスタに代入するのは、表示したキャラクタを消すために、Aレジスタに空白のキャラクタコード00Hを代入するからです。レジスタも変数とおなじで、あとから何かを代入すると、さきに代入したものが消えてしまうため、繰り返えし表示するキャラクタを代入するのです。

Aレジスタに表示するキャラクタを代入したら、そのキャラクタを画面に表示します。メモリ番地は、HLレジスタに代入してありますから、HLレジスタにAレジスタの内容を代入することになります。

```
LD (HL), A
```

キャラクタを画面に表示したら、ウェイト・ループを書いてもよさそうなのですが、そのまえにやることがあります。

PUSH命令で、Bレジスタの内容を退避させる

やることというのは、3行目でBレジスタに代入した繰り返えしの回数を、スタックに退避させることです。退避させる理由は、まえに説明したように、ウェイト・ループでもDJNZ命令を使っているからです。DJNZ命令の繰り返えしの回数は、Bレジスタに記憶させることになっていますから、2つのDJNZ命令で、ひとつのBレジスタを使うことになります。

レジスタは、変数とおなじと説明しました。したがって、同時にふたつの値を記憶することができません。また、DJNZ命令は繰り返えすたびにBレジスタから1を引いていきますから、キャラクタをAレジスタに繰り返えし代入す

るようなわけにはいきません。そこで、**PUSH**命令を使って、スタックに退避させるわけです。

PUSH命令は、つぎのような形をしています。

PUSH XX

XXには、**AF**、**BC**、**DE**、**HL**といったレジスタペアを置きます。

このプログラムの場合、**B**レジスタの内容を**PUSH**命令で退避させるわけですから、**PUSH**命令のXXのところに**BC**と置くことになります。

PUSH BC ————— **Bレジスタの内容をスタックに退避させる**

すると、**PUSH**命令によって、**B**レジスタの内容がスタックに退避されます。スタックは、**RAM**のスタック領域のなかに取りられます。

ウェイト・ループを入れる

PUSH命令で、**B**レジスタの内容をスタックに退避させたら、ウェイト・ループです。ここでは、まえの**FFH**を**64H**（10進数で100）にしています。

```
LD  B, 64H
LD  A, 64H ← 64H (10進数で100) 回繰り返す
DEC  A
JP  NZ  D010H
DJNZ D00EH
```

POP命令で退避させたBレジスタの内容を取り出す

さて、**B**レジスタの内容を、**PUSH**命令でスタックに退避させました。そこで、そのあとにある**DJNZ**命令のために、スタックに退避させた**B**レジスタの内容を取り出します。スタックに退避させた内容を取り出すのは、**POP**命令です。

POP命令は、つぎのような形をしています。

POP XX

XXのところには、**PUSH**命令で使ったレジスタペアを置きます。このプロ

グラムの場合は、**PUSH BC**としましたから、**POP BC**となります。

これで、スタックに退避させた**B**レジスタの内容が取り出されてきて、**B**レジスタの内容となります。

XOR A命令で、空白のキャラクタコードを作る

画面に表示されているキャラクタを消すために、**XOR A**命令で、空白のキャラクタコードを作ります（167頁参照）。

XOR A命令で作り出された**00H**は、**A**レジスタに記憶されています。したがって、画面に表示されているキャラクタを消すには、**A**レジスタの内容をメモリ番地を記憶している**HL**レジスタに代入してやります。

LD (HL), A

これで、画面に表示されているキャラクタが消えましたから、つぎのキャラクタを表示する、**VRAM**のメモリ番地を計算して出します。

キャラクタを表示しているメモリ番地-120

120は、16進数で**78H**で、1バイト（8ビット）の数です。メモリ番地は、**FC88H**というように2バイトの数です。1バイト（16ビット）の数と2バイトの数で、計算することはできませんから、**78H**の頭に**00H**をつけて、**0078H**という2バイトの数にします。メモリ番地から**0078H**を引くために、**LD**命令で**DE**レジスタに代入します。

LD DE, 0078H

2バイト（16ビット）の数の引き算の命令は、**SBC**命令ですから、**SBC**命令で、**HL**レジスタの内容から**DE**レジスタの内容を引きます。

SBC HL, DE

この計算結果は、**HL**レジスタに記憶されて残ります。

さて、最初のほうで説明した**DJNZ**がこのあとにきます。

DJNZ D008H

DJNZ命令で、繰り返し処理をする命令に戻す場合の指定は、相対アドレス指定です（145頁参照）。**DJNZ**命令で、繰り返し処理をする命令に戻るとき、**B**レジスタから1を引いて戻ります。**B**レジスタの内容が0になると、繰り返し処理をやめてつぎの命令の実行に移ります。

JP 5C66Hで、マシン語のコマンドを受けつける*印を表示します。

キャラクタの移動については、165頁を参照してください。

マシン語のプログラム

つぎに、アセンブルしたプログラムを示します。

メモリ番地	マシン語	アセンブリ言語
D000	21 88 FC	LD HL, FC88H
D003	11 78 00	LD DE, 0078H
D006	06 15	LD B, 15H
D008	3E EC	LD A, ECH
D00A	77	LD (HL), A
D00B	C5	PUSH BC
D00C	06 96	LD B, 96H
D00E	3E 96	LD A, 96H
D010	3D	DEC A
D011	C2 10 D0	JP NZ D010H
D014	10 F8	DJNZ D00EH
D016	C1	POP BC
D017	AF	XOR A
D018	77	LD (HL), A
D019	ED 52	SBC HL, DE
D01B	10 EB	DJNZ D008H
D01D	C3 66 5C	JP 5C66H

アセンブルの仕方 (例としてアセンブリ言語とマシン語の関係)

LD A, 50H=LD A, ×とn→3E n (nには50H)

LD HL, FC88H=LD HL, ×とnn→21 nn (nnにはFC88)

LD (HL), A=LD (HL)とA→77

PUSH BC=PUSH ×とBC→C5

DEC A=DEC ×とA→3D

JP NZ D010H=JP ×, nnとNZ→C2 nn (nnはD010H)

DJNZ D008H=DJNZ e→10 e-2 (e-2は相対アドレス)

SBC HL, DE=SBC HL, ×とDE→ED 52

XOR A=XOR ×とA→AF

キャラクタを画面の上から下へ移動する

画面の上から下にキャラクタを移動する場合も、画面の下から上にキャラクタを移動する場合もおなじです。違うところは、キャラクタを表示する最初のVRAMのメモリ番地を画面の下ではなくて、画面の上にとることです。

もうひとつは、つぎのメモリ番地の計算が引き算ではなくて、足し算になることです。

ウェイト・ループの説明、それにとまって追加される命令の説明、画面の上から下にキャラクタを移動させる命令の説明(172頁)など、すべて終わっていますから、つぎに参考までに、マシン語のプログラムを示すにとどめます。

なお、画面設定は、縦25行に変更して実行してください。

メモリ番地	マシン語	アセンブリ言語
D000	21 28 F3	LD HL, F328H
D003	11 78 00	LD DE, 0078H
D006	06 15	LD B, 15H
D008	3E EC	LD A, ECH
D00A	77	LD (HL), A
D00B	C5	PUSH BC
D00C	06 64	LD B, 64H
D00E	3E 64	LD A, 64H
D010	3D	DEC A
D011	C2 10 D0	JP NZ D010H
D014	10 F8	DJNZ D00EH
D016	C1	POP BC
D017	AF	XOR A
D018	77	LD (HL), A
D019	19	ADD HL, DE
D01A	10 EC	DJNZ D008H
D01C	C3 66 5C	JP 5C66H

画面に表示されているもの すべてを消す

比較命令CPを使って

ベーシックで画面を消去するときは、CLSあるいはPRINT CHR\$(12)を使います。この命令の場合は、ファンクションキーの内容の表示は消えません。ファンクションキーの内容の表示を消すときはCONSOLE,,0としなければなりません。マシン語で、画面に表示されているものをすべて消す場合、文字通り、画面に表示されているものすべてを消すことができます。ファンクションキーの内容の表示をもです。

さて、この画面消去では、**CP命令**という比較命令を使っています。

CP命令は、つぎのような形をしています。

CP n

CP命令のあとにあるのは、**n**ですから、**1**バイト（**8**ビット）の数値を置きます。**2**バイト（**16**ビット）の数値を置くことはできません。数値は、当然**16**進数です。たとえば、CP命令のあとに、

CP 06H

06Hを置きます。すると、**A**レジスタの内容と**06H**を比較し、その結果をゼロ・フラグに入れます。

Aレジスタの内容=06H

が等しいとき、ゼロ・フラグは、

ゼロ・フラグ=1

1になります。**1**になることを、ふつう**1が立つ**といいます。

このように、**A**レジスタの内容とある値とを比較するのが、CP命令の働きです。ゼロ・フラグに**1**が立ったあとどうするかは、**180**頁で説明したJP NZ命令を使います。

CP命令とJP NZ命令の具体的な働きについての説明は、画面消去のプログラムのなかで行うことにします。

VRAMのメモリ番地を指定して、00Hを表示する

画面に表示されているものをすべて消すには、空白のキャラクタコード00Hを、画面全体に表示させればよいのです。00Hを画面全体に表示するには、VRAMのメモリ番地を指定して行います。ただし、この場合はVRAMのメモリ番地内だけでなく、アトリビュートエリアのなかまで00Hを表示していく方法です。そうしないと、ファンクションキーの内容を表示している枠が消去されないからです。したがって、00Hを表示する最初のメモリ番地はVRAMのF300H、最後のメモリ番地はアトリビュートエリアのFEB7H、ということになります。

まず、VRAMの最初のメモリ番地F300Hを、LD命令でHLレジスタに代入します。

LD HL, F300H ← VRAMの最初のメモリ番地を代入

つぎに、画面に表示する空白のキャラクタコード00Hを、LD命令でAレジスタに代入します。

LD A, 00H ← 空白のキャラクタコードを代入

Aレジスタの内容(00H)を画面に表示するために、LD命令で、VRAMのメモリ番地を記憶しているHLレジスタに代入します。

LD (HL), A ← HLレジスタに、Aレジスタの内容00Hを代入

これで、最初の00Hが、メモリ番地F300Hに表示されます。

つぎのメモリ番地に00Hを表示するためには、HLレジスタの内容を変えなければなりません。VRAMのメモリ番地は、画面の右に行くにしたがってひとつずつ大きくなっています。基本的には、VRAMのメモリ番地は横80桁、縦25行で考えます。

そこで、1をプラスする命令INC命令(123頁参照)を使って、HLレジスタの内容に1をプラスします。

INC HL

この命令が実行されると、INC命令は1をプラスする命令ですから、HLレジスタの内容がF300Hの場合

F300H + 1 = F301H

となります。

F301Hは、HLレジスタに記憶されて残ります。

HLレジスタの数値の記憶の仕方

さて、このあとにでてくる比較命令CPのために、HLレジスタの数値の記憶の仕方について説明しておかなければなりません。HLレジスタは、

Hレジスタ=上位ビット（上位2桁）を記憶

Lレジスタ=下位ビット（下位2桁）を記憶

というように記憶しています。F301Hの場合を例にとると、上位ビットがF3H、下位ビットが01Hですから、

Hレジスタ=F3H

Lレジスタ=01H

というように記憶しているわけです。

CP命令で、上位ビットがFEHになったかを比較

さて、Hレジスタの内容を、Aレジスタに代入します。

LD A, H ← 上位ビットをAレジスタに代入

これは、CP命令で、最後のメモリ番地FEB7Hになったかどうかを比較するためです。ほんとうのところ、ベーシックのように、

IF HL=FEB7 THEN END

としてHLレジスタの内容が最後のメモリ番地になったかどうかを比較判定させ、最後のメモリ番地FEB7Hになったなら、00Hを画面全体に表示したことになるので終わり、というようにできると非常に便利です。

ところがマシン語では、このようなことはできません。まず、CP命令で比較します。このCP命令は、Aレジスタの内容と比較する、と決まっています。Aレジスタは8ビットレジスタですから、16ビットのメモリ番地全部を代入することができません。そこでまず、上位ビットを記憶しているHレジスタの内容を、LD命令でAレジスタに代入して、メモリ番地の上位ビットFEHになったかどうかを比較させます。Aレジスタの内容がFEHになったかどうか比較させるためには、CP命令のあとにFEHを置きます。

CP FEH ← CP命令でAレジスタの内容がFEHになったかどうか比較する

このようにすると、Aレジスタの内容がFEHと等しくなると、CP命令の働きによって、ゼロ・フラグに1が立ちます。Aレジスタの内容とFEHが等しくならないうちは、ゼロ・フラグは0になっています。

残念なことに、CP命令は比較するという働きなので、CP命令の働きはここまでです。つまり、ゼロ・フラグに1を立たせるか、立たせないかということまでなのです。その後の働きは、ほかの命令にたよらなければなりません。

JP NZ命令で判定して、その後の行動をする

Aレジスタの内容がFEHと等しくならなければ、まだ画面全体に00Hが表示されていないわけですから、00Hの表示を続けなければなりません。

ゼロ・フラグが0であるか1であるかを判定して、ゼロ・フラグが0のとき指定されたメモリ番地へジャンプする命令があります。180頁で取りあげたJP NZ命令です。そこで、JP NZ命令を使います。

JP NZ <ジャンプする先のメモリ番地>

ゼロ・フラグが0のうち、JP NZ命令がジャンプするときは、

LD A, 00H ← JP NZ命令でジャンプするさきの命令

です。なぜ、この命令まで戻すのかというと、途中でAレジスタにHレジスタの内容を代入しているからです。したがって、あらたに00HをAレジスタに代入する必要があるからです。

CP命令で、Aレジスタの内容とFEHが等しくなると、ゼロ・フラグに1が立ちます。ゼロ・フラグに1が立つと、JP NZ命令による指定されたメモリ番地へのジャンプは中止されて、つぎの命令の実行に進みます。

このあとにJP 5C66H、つまりふたたびマシン語のコマンドを受けつける命令を置いて実行させても、画面に表示されているものはすべて消されます。ただ、画面消去ができたからといって、それでいいというわけにはいきません。つぎは、CP命令でメモリ番地の下位ビットの比較を行います。

CP命令で、下位ビットがB7Hになったかを比較

00Hは、最後のメモリ番地FEB7Hまで表示させなければなりません。それには、メモリ番地の上位ビットを記憶しているHレジスタと、最後のメモリ番地の上位ビットFEHの比較だけでなく、メモリ番地の下位ビットを記憶しているLレジスタと、最後のメモリ番地の下位ビットB7Hとを、CP命令で比較します。まえに説明したように、CP命令は、Aレジスタとの比較を行うと決まっていますから、Lレジスタの内容をAレジスタに代入します。

LD A, L ← 下位ビットをAレジスタに代入

そして、CP命令のあとに比較する値を置きます。

CP B7H ← CP命令で、Aレジスタの内容がB7Hになったかどうか比較する

Aレジスタの内容がB7Hと等しくなると、ゼロ・フラグに1が立ちますが、等しくならないうちは、ゼロ・フラグは0となっています。

ゼロ・フラグが0のうちは、まだ00Hが最後のメモリ番地まで表示されていませんから、JP NZ命令で、

JP NZ D003H

まで戻します。Aレジスタの内容とB7Hが等しくなると、ゼロ・フラグに1が立ちます。ゼロ・フラグに1が立つと、JP NZ命令によるジャンプは終了します。これで、メモリ番地の上位ビットFEH、下位ビットB7Hまで00Hの表示ができた、つまり画面全体の消去が終わったわけです。

ふつうは、このあとにプログラムが続くわけですが、ここでは、JP命令でメモリ番地5C66Hへジャンプさせます。

では、つぎにアセンブルしたプログラムを示します。

メモリ番地	マシン語	アセンブリ言語
D000	21 00 F3	LD HL, F300H
D003	3E 00	LD A, 00H
D005	77	LD (HL), A
D006	23	INC HL
D007	7C	LD A, H
D008	FE FE	CP FEH
D00A	C2 03 D0	JP NZ D003H
D00D	7D	LD A, L
D00E	FE B7	CP B7H
D010	C2 03 D0	JP NZ D003H
D013	C3 66 5C	JP 5C66H

アセンブルの仕方 (例を参考に、アセンブル番地をみてください)

INC HL=INC ×とHL→23
LD A, H=LD A, ×とH→7C
CP FEH=CP ×とn→FE n (nにはFEH)

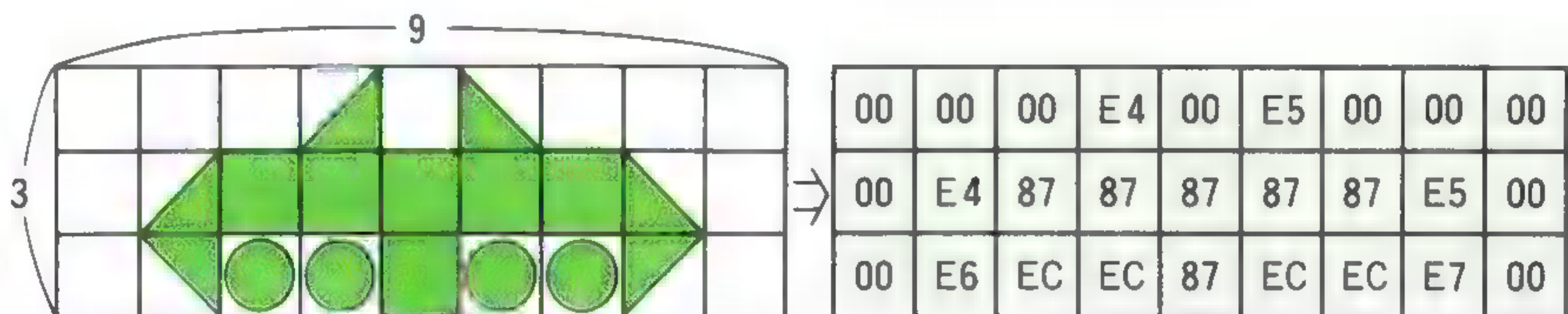
UFOやタンクなどの キャラクタを作るには

グラフィック・シンボルとキャラクタコードになおす

ゲーム・プログラムでは、UFOなどが飛んだり、タンクなどが走り回ったりしています。ここでは、タンクを取りあげて、キャラクタの作り方について説明することにします。

195頁で、ここで説明する方法で作ったタンクの表示の仕方を説明します。また206頁では、弾を発射する方法について説明することにします。

ここで作るタンクは、つぎに示す図のような形のものです。



まず最初に、自分で表示したいキャラクタを、うえに示した図のように、グラフィック・シンボルで書きます。

ところで、うえに示した図をみると、タンクの左右に空白が取ってあります。この空白は、タンクが左右に移動したとき、あとに表示されているキャラクタの一部分を消すためのものです。この空白がないと、キャラクタの一部分が尾を引いてしまうことになります。

キャラクタが左右に移動するものではなくて、右へ向かって移動するときは、左側に空白を取ります。左に向かってだけ移動するときは、逆に右側に空白を取ります。当然のことですが、移動しないときは、空白は必要ありません。

このように、自分で表示したいキャラクタをグラフィック・シンボルで書いたら、つぎは、それぞれのグラフィック・シンボルが持っているキャラクタコードになおします。キャラクタコードになおすのは10進数ではなく、16進数でなおします。

空白のキャラクタコードは、00Hです。したがって、空白のところはすべて00Hと書きます。

▲は、キャラクタコードE4Hですから、E4Hと書きます。■は、E5Hですから、E5Hと書きます。このように16進数でキャラクタコードを書いていくと、前頁の図の右に示したような表ができあがります。

この表に示したキャラクタコードを、マシン語のプログラムのなかにデータとして設定しておいて、ひとつずつ読み取らせて、画面に表示していきます。

キャラクタコードをデータとして設定して、ひとつひとつ読み取らせ、画面に表示する方法については、195頁を参照してください。



タンクなどのキャラクタを画面に表示する

キャラクタのデータは、LD A, (DE) で読み取る

193頁でタンクを作って、自分が表示したいキャラクタの作り方を説明しましたが、キャラクタの作り方は、意外と簡単だったと思います。

ではここでは、自分で作ったキャラクタの表示の仕方を、タンクの表示の仕方を例にして説明することにします。

193頁で、タンクをグラフィック・シンボルで書きました。そして、それをそれぞれのキャラクタが持っている16進数のキャラクタコードになおしました。その16進数のキャラクタコードが、データとなります。

このデータを設定するメモリ番地は、HALT（ベーシックのEND命令とおなじ）のあとのメモリ番地からとなります。

たとえば、ベーシックでREAD～DATA命令を使うとき、END命令のあとにDATA命令を置いて、そこにデータを設定します。マシン語では、DATA命令はありませんが、それとおなじように、HALT命令のあとのメモリ番地に記憶させるわけです。このキャラクタを表示するプログラムでは、データはつぎに示すように、メモリ番地D020Hから割り当てられます。

D020	00	00	00	E4	00	E5	00	00	00
D029	00	E4	87	87	87	87	87	E5	00
D032	00	E6	EC	EC	87	EC	EC	E7	00

おことわりしておきますが、このデータを記憶させるメモリ番地は、適当にD020Hを選んで、割り当てているわけではありません。このキャラクタを表示するプログラムはすでにできあがっていて、プログラムの最初のメモリ番地をD000Hに割り当てているため、データのメモリ番地がD020Hになっているわけです。

キャラクタを表示するプログラムをわかりやすく説明するために、説明の順序が逆になっているのです。

データの最初のメモリ番地をレジスタに代入

つぎに、メモリ番地D020Hから設定したデータを読み取らせます。データを読み取らせるには、まず、LD命令を使って、データに割り当てた最初のメモリ番地D020Hをレジスタに代入します。この場合、メモリ番地は2バイト（16ビット）なので、レジスタペアを使います。つまり、BCレジスタとかDEレジスタです。HLレジスタもありますが、これはキャラクタを画面に表示する、VRAMのメモリ番地を記憶させるために使いますから、ここでは除いておきます。そこで、DEレジスタを使うことにします。

LD DE, D020H データに割り当てた最初のメモリ番地を代入

そして、DEレジスタに代入したデータの最初のメモリ番地を、Aレジスタに代入します。Aレジスタに最初のメモリ番地を代入する、と説明するとLD命令を使って、

LD A, DE

というようにすればよいと考えるかもしれませんが、Aレジスタは1バイト（8ビット）のレジスタなので、2バイト（16ビット）のDEレジスタの内容を代入することはできません。

データを読み取らせるには

ここで、DEレジスタに代入したメモリ番地をAレジスタに代入するということは、DEレジスタに代入したメモリ番地そのものではなく、DEレジスタのメモリ番地で指定したデータをAレジスタに代入する、ということです。そのようにするには、つぎに示すように、DEレジスタをカッコで囲んで、Aレジスタの右側に置きます。

LD A, (DE)

このようにすると、DEレジスタのメモリ番地そのものではなく、そのメモリ番地で指定したデータが、Aレジスタに代入されることになります。

さてこれで、DEレジスタのメモリ番地で指定したデータがAレジスタに代入されましたから、つぎは、VRAMのメモリ番地を指定して、データを表示させます。

VRAMのメモリ番地は、HLレジスタに代入します。HLレジスタには、デ

タンクなどのキャラクタを画面に表示する

ータを表示する最初のメモリ番地を指定します。この場合は、メモリ番地F9AFHを代入します。

```
LD HL, F9AFH
```

そしてAレジスタのデータをHLレジスタに代入して、画面に表示します。

```
LD (HL), A
```

これで、1行目の1個目のデータ00Hが画面に表示されます。

さて、データをつぎつぎに読み取らせて画面に表示させていくには、命令を追加しなければなりません。そこで、データの部分を除いて、これまで説明した命令をまとめておきます。

```
LD HL, F9AFH  データを表示するVRAMのメモリ番地を代入
```

```
LD DE, D020H  データの最初のメモリ番地を代入
```

```
LD A, (DE)  指定したメモリ番地のデータをAレジスタに代入
```

```
LD (HL), A  データを画面に表示する
```

データの読み取りと、表示を繰り返す

データは、1個ではありません。そこで、データの読み取りと、データの表示を繰り返す必要があります。

データの読み取りを繰り返して行うには、DEレジスタの内容を、ひとつずつ増やしていく必要があります。なぜなら、データは、

```
D020=00H    D021=00H    D022=00H
```

```
D023=E4H    D024=00H    D025=E5H
```

というように記憶されているからです。また、表示するメモリ番地も、最初にF9AFHにデータを表示しましたから、つぎはF9B0H、そのつぎはF9B1Hというように、ひとつずつ増やしていかなければなりません。

1を加算する命令は、INC命令です。INC命令で、DEレジスタの内容と、HLレジスタの内容に1をプラスするには、

```
INC DE  DEレジスタの内容に1プラス
```

```
INC HL  HLレジスタの内容に1プラス
```

となります。

このINC命令は、1をプラスする命令ですから、このINC命令を含めて繰り返さなければなりません。

繰り返す回数は、1行に9個データが並んでいるので、9回繰り返します。

なぜ、1度にデータ全部の個数だけ繰り返えさないのかというと、1行目はおなじ列に並ぶので、1プラスするだけですみますが、2行目はその列の下、3行目はまたその列の下、というように表示するメモリ番地を変えていかなければならないからです。

さて、9回繰り返えすには、CP命令とJP NZ命令を使います。繰り返えし専門の命令には、DJNZ命令がありますが、この命令は、このあとに使うので、ここでは使いません。

カウントする命令を置く

この場合はJP NZ命令(180頁参照)で繰り返えす回数は、9回です。そこで、JP NZ命令のまえに、回数をカウントする命令を置きます。ベーシックの場合、9という数をカウントさせて、9にならないうちはなにかをさせるときには、

C = C + 1

IF C <> 9 THEN 行番号

というように書きますが、これとおなじようなことをするわけです。

カウントさせるには、まず繰り返えしを始める最初の数値を、レジスタに代入します。ここでは、Cレジスタにします。最初の数値は1にします。16進数では01Hです。

LD C, 01H

つぎに、Cレジスタの01Hを、02H、03Hというように1ずつ増やしていきます。1ずつ加算していく命令は、INC命令ですから、

INC C

となります。Cレジスタの内容を、01Hから始めて9回繰り返えして1ずつプラスしていくと、0AH(10進数で10)になりますから、Cレジスタの内容が0AHになったかどうか、比較する命令CP命令で比較させます(188頁参照)。ところで、CP命令はAレジスタの内容と、CP命令のあとに置かれた1バイトの数値とを比較すると決まっているので、まず、Cレジスタの内容をAレジスタに代入しておかなければなりません。

LD A, C ← Cレジスタの内容をAレジスタに代入

そのうえで、CP命令のあとに0AHを置いて比較させます。

CP 0AH ← CP命令で、Aレジスタの内容と0AHを比較

タンクなどのキャラクタを画面に表示する

このようにすると、C P命令は、Aレジスタの内容と0 A Hを比較していつて、Aレジスタの内容が0 A Hと等しくなったとき、C P命令はゼロ・フラグに1を立たせます。Aレジスタの内容が0 A Hと等しくない間は、ゼロ・フラグは0です。

さきに取りあげたJ P N Z命令は、ゼロ・フラグがゼロのとき、J P命令に指定されたメモリ番地にジャンプします。

J P N Z <L D A, (D E) 命令があるメモリ番地を置く>

この場合は、J P N Z命令のあとには、L D A, (D E)のメモリ番地を置きます。つまり、ゼロ・フラグが0のうち、このメモリ番地にジャンプして9回繰り返えすと、つぎの頁の実行結果に示すように、9個のデータがAレジスタに読み取られて、H Lレジスタで指定したV R A Mのメモリ番地に表示されるわけです。

比較命令C P命令の働きで、ゼロ・フラグに1が立つと、J P N Z命令はジャンプを終えて、つぎの命令の実行に移ります。

では、これまで説明した命令をまとめて、実行結果を示します。プログラムを完成するには、まだ命令を追加しなければなりません。したがって、マシン語にふられるメモリ番地は、D 0 0 2番地からとなっています。また、命令が抜けているところには、0 0 Hを入れています。このようにしたのは、J P命令などジャンプする先が違ってしまうと、プログラムが実行されないからです。したがって、このプログラムを実行させるときは、メモリ番地をD 0 0 2 Hから入力し、0 0 Hも必ず入力してください。このプログラムの最後は、J P 5 C 6 6 Hで、ふたたびマシン語のコマンドを受けつけるようにしています。

なお画面設定は、ダイレクト・モードで、

W I D T H 80, 25

に設定して実行してください。このように画面設定を80桁×25行に変更しないと、キャラクタは表示されません。

メモリ番地	マシン語	アセンブリ言語
D 0 0 2	2 1 A F F 9	L D H L, F 9 A F H
D 0 0 5	1 1 2 0 D 0	L D D E, D 0 2 0 H
D 0 0 8	0 0	
D 0 0 9	0 0	

D00A	0E 01	LD C, 01H
D00C	1A	LD A, (DE)
D00D	77	LD (HL), A
D00E	23	INC HL
D00F	13	INC DE
D010	0C	INC C
D011	79	LD A, C
D012	FE 0A	CP 0AH
D014	C2 0C D0	JP NZ D00CH
D017	C3 66 5C	JP 5C66H
D020	00 00 00	
D023	E4 00 E5	
D026	00 00 00	

つぎに示すのが、実行結果です。

実行結果

MON

#50002

3個空白を表示

▲▲▲

PUSH命令でHLレジスタ、Bレジスタの内容を退避

さて、1行分のデータを読み取って、画面に表示しました。データは3行あります。したがって、さきにまとめた命令を3回繰り返えさなければ、キャラクタ全体を表示することができません。

さきにまとめた命令を3回繰り返えすには、繰り返えしの命令DJNZ命令を使います。DJNZ命令はプログラムの最後に置くことになりましたが、ここでDJNZ命令を取りあげたのは、DJNZ命令を使う場合、繰り返えす回数をBレジスタにセットしなければならないからです。この場合、DJNZ命令で繰り返えす回数は3回ですから、16進数03HをBレジスタに代入します。

LD B, 03H ← 繰り返えしの回数を代入

この命令は、さきにまとめた命令の一番最初に置きます。

LD B, 03H

LD HL, F9AFH

LD DE, D020H

と、いうようにです。

この命令のあとに、PUSH命令を使って、Bレジスタに代入した繰り返えしの回数03Hを、スタックに退避させます。

PUSH BC ← Bレジスタの内容をスタックに退避させる

PUSH命令については183頁を参照してもらうことにして、ここでPUSH命令で、Bレジスタの繰り返えしの回数03Hをスタックに退避させるのは、あとで、BレジスタをBCレジスタペアにして使わなければならないからです。このことはあとでわかります。おなじようにHLレジスタの内容も、PUSH命令を使ってスタックに退避させます。

PUSH HL ← HLレジスタの内容をスタックに退避させる

なぜ、HLレジスタの内容をスタックに退避させるのかというと、HLレジスタの内容は、つぎのように、

F9AFH+0078H ← 120を足して2行目のメモリ番地を指定

FA27H+0078H ← 120を足して3行目のメモリ番地を指定

HLレジスタの内容に0078H（10進数の120）を足して、2行目、3行目にキャラクタを表示させる、メモリ番地の計算に使わなければなりません。

ところが、HLレジスタの内容は、もうひとつのところで使わなければなり

ません。それは、さきにまとめた命令のなかで説明した、

INC HL

です。ここで、1回目にINC命令で9回繰り返えすと、メモリ番地は、

F9B8H

になってしまって、2行目のメモリ番地の指定の計算には役にたたなくなってしまう。そこで、PUSH命令で、もとのHLレジスタの内容が壊されないように、スタックに退避させるわけです。

このふたつのPUSH命令は、さきにまとめた命令のなかで、00Hを入れたところがありますが、そのところに入ります。

さて、さきにまとめた命令を実行させるために、JP 5C66命令を入れましたが、その命令は取り去って、そのあとから、これから説明する命令が入ります。

2行目、3行目にキャラクタを表示するメモリ番地の計算

さきにBレジスタの内容をPUSH命令で退避させました。それは、ここでキャラクタを表示する、2行目、3行目の表示位置を指定するメモリ番地の計算をするために、Bレジスタを使うからです。もちろん、メモリ番地は2バイト（16ビット）ですから、Bレジスタひとつではありません。BCレジスタペアにして使います。それでもBレジスタの内容は破壊されてしまいます。そこで、PUSH命令で退避させたわけです。

さてキャラクタを表示する、2行目、3行目の表示位置を指定するメモリ番地の計算のために、0078H（10進数で120）を、LD命令でBCレジスタに代入します。

LD BC, 0078H

キャラクタを表示する2行目、3行目の表示位置を計算するために、なぜ0078Hを使うかは151頁を参照してください。

BCレジスタに0078Hを代入したら、POP命令でスタックに退避させたメモリ番地をHLレジスタに取り出します。

POP HL

1回目にPUSH命令で退避させたメモリ番地はF9AFHですから、POP命令で取り出されてくるメモリ番地はF9AFHです。

そこでHLレジスタの内容とBCレジスタの内容をADD命令で加算します。

ADD HL, BC

HLレジスタの内容はF9AFH、BCレジスタの内容は0078Hですから、つぎのような足し算が行われます。

F9AFH+0078=FA27H

加算した結果のFA27Hが、キャラクタを表示する2行目の最初のメモリ番地です。巻末に示した80桁×25行モードのメモリ番地表をみるとFA27Hは、最初のF9AFHの真下のメモリ番地であることがわかんと思います。

この加算結果のFA27Hは、HLレジスタに記憶されて残ります。

DJNZ命令で、繰り返し処理をする最初の命令に戻る

つぎは、DJNZ命令で繰り返し処理をするため、繰り返し処理をする最初の処理に戻りますが、そのまえにPUSH命令で、退避させたBレジスタの内容03Hを取り出してこなければなりません。なぜなら、DJNZ命令で繰り返し処理をするために戻るとき、Bレジスタから1を引いて戻るからです。

POP BC ← スタックに退避させた内容を取り出す

POP BC命令で取り出されてきた03Hは、BCレジスタのBレジスタに記憶されています。そこで、DJNZ命令で戻るとき、Bレジスタの03Hから1を引くことができるわけです。DJNZ命令で戻る最初の命令は、PUSH BC命令です。これは、BCレジスタに記憶されている03H-1Hの結果の02Hが破壊されないためです。このことは、すでに説明しました。

おなじようにPUSH HL命令で、HLレジスタに記憶されているF9AFH+0078Hの計算結果のFA27Hが破壊されないためにスタックに退避させます。

あとは、まえに説明した通りですが、DJNZ命令で戻ってきた2回目についておおまかに説明すると、DEレジスタにはD029Hが記憶されているので、LD A, (DE)でD029Hからのデータの読み取りを始めます。HLレジスタにはFA27Hが記憶されていますから、VRAMのメモリ番地FA27Hからキャラクタの表示を始めることになります。

3回目のDEレジスタの内容はD032Hになります。HLレジスタの内容はFA9FHになります。

Bレジスタの内容が0になると、DJNZ命令の繰り返し処理は終了して、つぎの命令の実行に移ります。

アセンブルしたプログラム

では、つぎにこれまで説明した命令をまとめ、アセンブルしたプログラムを示します（画面は、80×25モード）。

メモリ番地	マシン語	アセンブリ言語
D000	06 03	LD B, 03H
D002	21 AF F9	LD HL, F9AFH
D005	11 20 D0	LD DE, D020H
D008	C5	PUSH BC
D009	E5	PUSH HL
D00A	0E 01	LD C, 01H
D00C	1A	LD A, (DE)
D00D	77	LD (HL), A
D00E	23	INC HL
D00F	13	INC DE
D010	0C	INC C
D011	79	LD A, C
D012	FE 0A	CP 0AH
D014	C2 0C D0	JP NZ D00CH
D017	01 78 00	LD BC, 0078H
D01A	E1	POP HL
D01B	09	ADD HL, BC
D01C	C1	POP BC
D01D	10 E9	DJNZ D008H
D01F	76	HALT

データ

D020	00 00 00 E4 00 E5 00 00 00
D029	00 E4 87 87 87 87 87 E5 00
D032	00 E6 EC EC 87 EC EC E7 00

タンクなどのキャラクタを画面に表示する

つぎに実行結果を示します。2 回目を実行すると、まえに示した実行結果の下に、そのデータのキャラクタが表示されます。



3 回目を実行すると、2 回目の実行結果の下に、3 行目のデータのキャラクタが表示されて完成です。



タンクから弾を発射する

キー入力は、IN命令

195頁でタンクを画面に表示しました。ただ、タンクを画面に表示しただけではおもしろくないので、新しい命令をおぼえるために、タンクから弾を発射させることにしましょう。当然のことですが、タンクから弾を発射するには、弾を発射させるキー入力を行います。このキー入力を受けつける命令が**IN命令**です。IN命令は、つぎのような形をしています。

IN X, (C)

IN命令のあとのXのところには、A、B、C、D、E、H、Lの各レジスタを置くことができますが、カッコのなかは、Cレジスタと決まっています。

このIN命令の働きは、ベーシックのINP関数の働きとおなじです。ベーシックのINP関数は、たとえばつぎのように、カッコのなかを、

A=INP (0)

とすると、0から7までのキー入力を受けつけます。

IN命令もおなじで、カッコのなかのCレジスタに0を与えると、0から7までのキー入力を受けつけます。カッコのなかのCレジスタに0を与えるには、IN命令のまえで、LD命令を使って、Cレジスタに0を代入すればよいのです。この場合は、IN命令のXのところにAレジスタを置きます。

LD C, 00H

IN A, (C)

これでIN命令は、0から7までのキー入力を受けつけます。

この場合、0から7までのキーを押したとき、IN命令で取り出されてきてAレジスタに代入されるデータは0から7という数ではなくて、0から7までのキーに対応した8ビットのデータです。この8ビットのデータは、INP関数を使った場合、変数Aに代入される値とおなじです。

たとえば、5のキーを押したときINP関数の場合は、10進数で考えて、223が

変数Aに代入されることになりますが、IN命令の場合は、16進数で考えて8ビットのDFHがAレジスタに代入されることになります。

では、つぎに0から7までのキーを押したときに、IN命令で取り出されてきて、Aレジスタに代入される8ビットのデータを示します。

0のキーを押すと、FEH（10進数で254）
1のキーを押すと、FDH（10進数で253）
2のキーを押すと、FBH（10進数で251）
3のキーを押すと、F7H（10進数で247）
4のキーを押すと、EFH（10進数で239）
5のキーを押すと、DFH（10進数で223）
6のキーを押すと、BFH（10進数で191）
7のキーを押すと、7FH（10進数で127）

このように、押されたキーの入力を受けつけて、8ビットのデータを取り出して、Aレジスタに代入するのがIN命令の働きです。

CP命令で比較する

さて、IN命令のカッコのなかのCレジスタに0を代入すると、0から7までのキーの入力ができるようになりますが、弾を発射するというような場合は、ふつう押すキーを限定します。

ベーシックでは、弾丸を発射するというような場合は、ふつうINKEY\$関数を使いますが、説明の都合上、INP関数で行います。たとえば、5のキーが押されたときに弾丸を発射する場合、つぎのように、IF～THEN文を使って、INP関数で取り出されて、変数Aに代入された値が223であるかどうかを判定して、実行させます。

A=INP (0)

IF A=223 THEN 350

IN命令の場合もおなじです。5のキーが押されたとき弾丸を発射するという場合、つぎに示すように、比較をするCP命令（188頁参照）を使って、IN命令で取り出されてきてAレジスタに代入された8ビットのデータが、DFH（10進

数で223)であるかどうか、比較させます。比較する値DFHは、CP命令のあとに置きます。

LD C, 00H ・—— Cレジスタに00H(10進数で0)を代入

IN A, (C) ・—— IN命令で、押されたキーのデータを取り出してAレジスタに代入

CP DFH ・—— CP命令で、Aレジスタの値がDFHかどうか比較する

CP命令は、Aレジスタの内容と比較すると決まっているので、IN命令を使うときは、取り出されてくる8ビットのデータをAレジスタに代入するようにしたほうが便利です。

さて、Aレジスタのデータと、CP命令に置かれているDFHとが等しいときCP命令の働きで、ゼロ・フラグに1が立ちます。AレジスタのデータとDFHと等しくないときは、ゼロ・フラグは0です。比較するCP命令の働きは、このように、Aレジスタの内容とある値と比較するという命令です。

ゼロ・フラグに1が立ったときに、弾を発射するところにジャンプするという、つぎの実行に移すには、もうひとつの命令が必要になります。

JP Z 命令は、ゼロ・フラグが1のときジャンプする

180頁で、JP NZ命令について説明していますが、JP NZ命令は、ゼロ・フラグが0のとき条件が成立して、JP NZ命令に示されているメモリ番地にジャンプします。

これに対して、JP Z命令は、ゼロ・フラグが1のとき条件が成立して、JP Z命令に示されているメモリ番地にジャンプします。弾丸を発射する場合は、JP Z命令のあとに、弾を発射する最初の命令のあるメモリ番地が置かれますから、そのメモリ番地にジャンプすることになります。

JP NZ ゼロ・フラグが0のとき指定されたメモリ番地にジャンプ

JP Z ゼロ・フラグが1のとき指定されたメモリ番地にジャンプ

JP命令のあとにNZがあるか、Zがあるかで、このように相違しますから注意して使わなければなりません。

さてここでは、キー入力があったときタンクから弾を発射させるだけにしますから、JP Z命令のあとにJP命令を置いて、キー入力があるまで、LD C, 00H命令にもどして、キー入力待ちの状態にします。

JP <LD C, 00H命令がある番地>

以上が、弾を発射するキー入力のコマンドです。つぎにまとめて示します。

```
LD  C, 00H
IN  A, (C)
CP  DFH
JP  Z  <弾を発射するメモリ番地>
JP  <LD C, 00H命令があるメモリ番地>
```

キー入力があるまで
キー入力待ちの状態になる

キー入力があると
ジャンプする

この弾丸を発射するコマンドは、つぎのタンクを表示するコマンドにつづきます。

弾を発射するタンクを表示する

弾丸をタンクから発射するために、まずタンクを画面に表示します。タンクを表示するプログラムは204頁で説明したプログラムとおなじです。違う点は弾を画面の上に向けて発射するために、タンクを表示するVRAMのメモリ番地を、つぎのようにFC80Hに下げただけなので、説明を省略します。

LD HL, FC80H

弾丸を発射する

弾丸を発射するプログラムのなかでは、新しく取りあげたという命令はありません。またプログラムが少し長いので、まず、アセンブリ言語で書いたプログラムを示し、そのあとで説明を加えていくことにします。

```
LD  B, 13H
LD  DE, 0004H
ADD HL, DE
LD  A, ECH
LD  (HL), A
PUSH BC
LD  B, F0H
LD  A, F0H
DEC A
JP  NZ <DEC Aのメモリ番地>
DJNZ <LD A, F0Hのメモリ番地>
```

ウェイト・ループ


```

POP BC
LD (HL), A
LD DE, 0078H
SBC HL, DE
DJNZ D056H
JP <タンクを表示させるD000番地>

```

では、それぞれの命令の働きについて説明します。

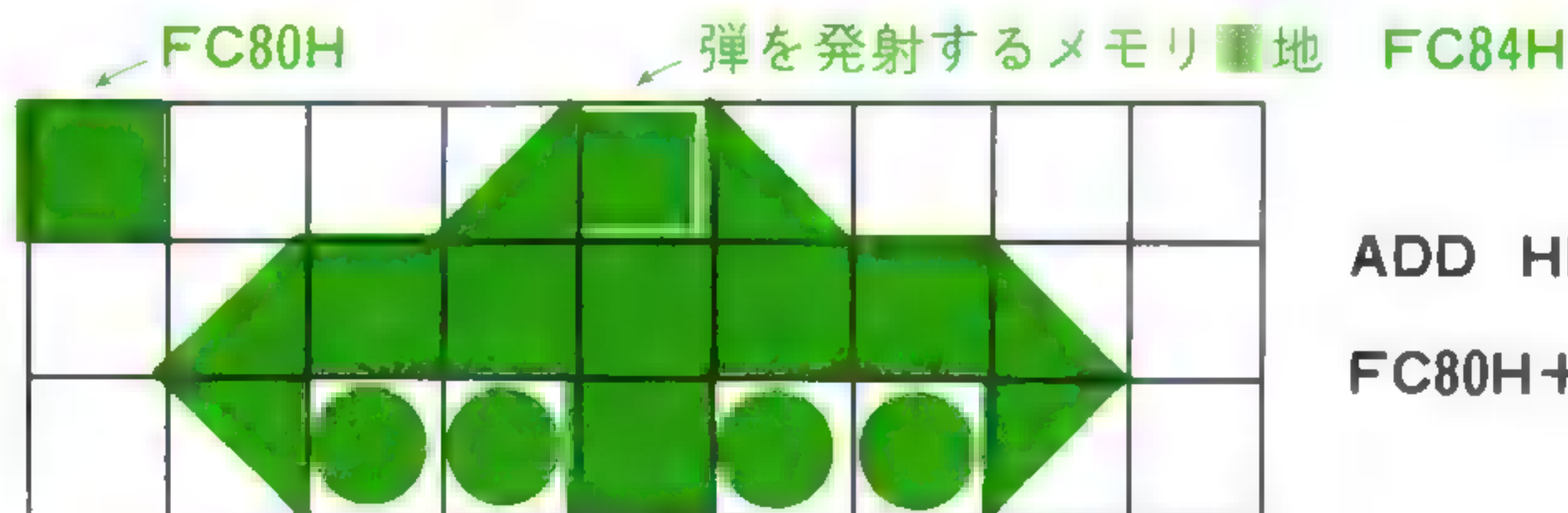
弾を発射するキー入力があると、このプログラムの最初の命令にジャンプしてきます。最初の命令は、

```
LD B, 13H
```

で、LD命令でBレジスタに13H（10進数の19）を代入します。弾をタンクから画面の上まで表示していくには、弾を表示したり、消したりということをして19回繰り返えして行わなければなりません。繰り返えしは、繰り返えし専門の命令であるDJNZ命令で行います。DJNZ命令で繰り返えす回数は、Bレジスタにセットしておくものと決まっていますから、まず、LD命令で、DJNZ命令で繰り返えす回数13H（10進数で19）を、Bレジスタに代入します。

```
LD DE, 0004H
```

は、LD命令で、DEレジスタに0004H（10進数で4）を代入します。なぜ、HLレジスタに0004Hを代入するのかというと、HLレジスタにはつぎの図に示すように、タンクを表示したVRAMのメモリ番地FC80Hが記憶されています。弾を発射する位置は、そのVRAMのメモリ番地より4番地右側によった位置だからです。



```
ADD HL, DE
FC80H + 0004H = FC84H
```

そこでDEレジスタに0004H（10進数で4）を代入して、HLレジスタのメモリ番地に0004Hをプラスし、その位置から弾を発射するわけです。

ADD HL, DE

で、さきにLD命令でDEレジスタに代入した0004Hと、HLレジスタに記憶されているVRAMのメモリ番地FC80HとをADD命令で加算します。この加算結果のFC84Hは、HLレジスタに記憶されて残り、弾を最初に表示するVRAMのメモリ番地となります。

LD A, ECH

で、画面に表示する●のキャラクタコードECHをAレジスタに代入します。ここで、Aレジスタに代入した●のキャラクタコードECHは、つぎにLD命令で、HLレジスタに代入されて、

LD (HL), A

画面に表示されます。HLレジスタの内容は、さきにADD命令で計算された結果のFC84Hです。

さて、●を表示して、すぐに●を消したのでは、命令が高速に処理されるため、●が画面に表示されません。そこで●を表示したあと、●を消す命令の実行を遅らせます。そのようにするには、178頁で説明しているウェイト・ループを使います。このウェイト・ループでも、DJNZ命令を使います。したがって、Bレジスタをも使うことになります。Bレジスタは、まえにも使っていますから、まえにBレジスタに記憶させた内容が破壊されないように、PUSH命令でスタックに退避させます。

PUSH BC

この場合、Bレジスタという1バイトのレジスタでは、スタックに退避させることはできません。そこで、BCレジスタペアにして退避させることになります。Bレジスタに記憶されていた内容を、PUSH命令でスタックに退避させたら、ウェイト・ループを置きます。

```
LD B, F0H
LD A, F0H
DEC A
JP NZ <DEC Aのメモリ番地>
DJNZ <LD A, F0Hのメモリ番地>
```

この場合は、LD命令でBレジスタにF0H（10進数で240）、LD命令でAレジスタにF0Hを代入していますから、最初、


```
LD  A, F0H
DEC A ←
JP  NZ <DEC Aのメモリ番地>
```

内側の命令を実行します。DEC命令は1を引く命令です。したがって、Aレジスタの内容が0になるまで、JP NZ命令でDEC A命令にジャンプして、DEC A命令を繰り返えします。Aレジスタの内容が0になると、ゼロ・フラグに1が立ちますから、JP NZ命令による繰り返えしはやめて、つぎの命令DJNZ命令を実行します。DJNZ命令は、BレジスタのF0Hから1を引いて、LD A, F0H命令を実行して、AレジスタにF0Hを代入しますから、さきに説明したことを、ふたたび繰り返えします。

このことを、Bレジスタの内容が0になるまで繰り返えすと、ウェイト・ループから抜け出して、画面に表示されている●を消すことになります。ウェイト・ループについては178頁を参照してください。

さて、表示されている●を消します。●を消すには、185頁で説明しているように、XOR A命令を使いますが、ここではウェイト・ループでAレジスタが00H（10進数で0）になっていますから、そのままAレジスタの内容をHLレジスタに代入することになります。

```
LD  (HL), A
```

これで、画面に表示されている●は消されます。

つぎは、つぎの●を表示するためのVRAMのメモリ番地の計算です。

VRAMのメモリ番地は、画面の上に行くにしたがって、メモリ番地が小さくなっていきますから、VRAMのメモリ番地を記憶しているHLレジスタから0078H（10進数で120）を引きます。なぜ、0078H（10進数で120）を引くかは185頁を参照してください。引き算の計算を行うためには、LD命令で0078HをDEレジスタに代入します。

```
LD  DE, 0078H
```

そのうえで、2バイト（16ビット）の引き算の命令SBCを使って、HLレジスタの内容からDEレジスタの内容を引きます。

```
SBC HL, DE
```

この引き算の結果は、HLレジスタに記憶されて残ります。そして、つぎの●を表示する、VRAMのメモリ番地の指定となります。

これで、●を表示して消すという命令は終わりです。そこで、このことを●が画面の上にたどりつくまで、つまり、最初にBレジスタにセットした13Hが0になるまで、繰り返えしの命令DJNZ命令で、繰り返えし処理する最初の命令が記憶されているメモリ番地に戻します。

DJNZ命令で、繰り返えしを始める最初の命令、ここでは、LD A, ECHに戻る時、Bレジスタから1を引いて戻りますから、Bレジスタから1を引くことができるように、PUSH BC命令でスタックに退避させたBレジスタの内容を、POP命令で取り出します。

POP BC

POP BC命令でスタックから、PUSH BC命令でスタックに退避させた内容を、BCレジスタに取り出すと、

DJNZ <LD A, ECHのメモリ番地>

その内容から1を引いて、DJNZ命令で、繰り返えし処理する最初の命令が記憶されているメモリ番地に戻ります。

DJNZ命令の繰り返えし処理する回数は、最初にBレジスタに代入した13H（10進数で19）回です。したがって、Bレジスタの13Hが0になると、DJNZによる繰り返えし処理は終了して、つぎの命令の実行に移ることになります。

DJNZ命令による繰り返えしが終了したということは、発射した弾が画面の上まで上がったということですから、ふたたび、タンクを表示させるD000番地に戻します。その命令に戻す命令は、JP命令です。JP命令のあとに、戻すべき命令が記憶されているメモリ番地を置くと、無条件にそのメモリ番地にジャンプします。

JP <タンクを表示させるD000番地>

これで、タンクから弾を発射するプログラムは終了です。

タンクを表示するデータ

この弾を発射するプログラムのあとに、タンクを画面に表示するデータを置きます。タンクを表示するデータは195頁で説明したのと、この場合はまったくおなじですので、説明は省略します。プログラムを実行するにはWIDTH 命令で80桁、25行表示にし、テンキーの[5]のキーを押してください。

では、つぎにアセンブルしたプログラムを示します。

アセンブルしたプログラム

メモリ番地	マシン語	アセンブリ言語
D000	06 03	LD B, 03H
D002	21 80 FC	LD HL, FC80H
D005	E5	PUSH HL
D006	11 80 D0	LD DE, D080H
D009	C5	PUSH BC
D00A	E5	PUSH HL
D00B	0E 01	LD C, 01H
D00D	1A	LD A, (DE)
D00E	77	LD (HL), A
D00F	23	INC HL
D010	13	INC DE
D011	0C	INC C
D012	79	LD A, C
D013	FE 0A	CP 0AH
D015	C2 0D D0	JP NZ D00DH
D018	E1	POP HL
D019	01 78 00	LD BC, 0078H
D01C	09	ADD HL, BC
D01D	C1	POP BC
D01E	10 E9	DJNZ D009H
D020	E1	POP HL
D021	0E 00	LD C, 00H
D023	ED 78	IN A, (C)
D025	FE DF	CP DFH
D027	CA 50 D0	JP Z D050H
D02A	C3 21 D0	JP D021H
D050	06 13	LD B, 13H
D052	11 04 00	LD DE, 0004H

タンク表示とキー入力

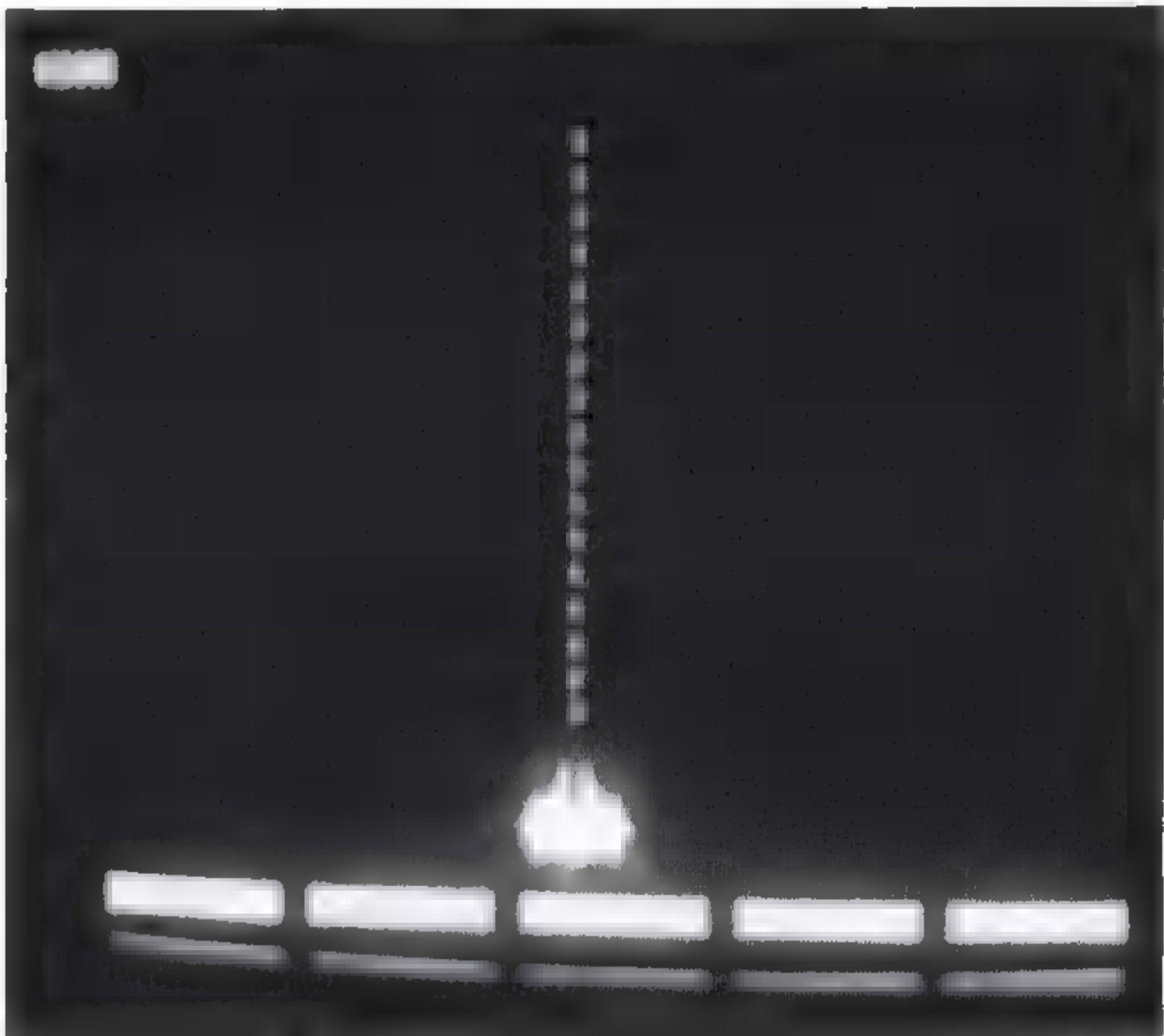
タンクから弾を発射する

D055	19	ADD HL, DE
D056	3E EC	LD A, ECH
D058	77	LD (HL), A
D059	C5	PUSH BC
D05A	06 F0	LD B, F0H
D05C	3E F0	LD A, F0H
D05E	3D	DEC A
D05F	C2 5E D0	JP NZ D05EH
D062	10 F8	DJNZ D05CH
D064	77	LD (HL), A
D065	11 78 00	LD DE, 0078H
D068	ED 52	SBC HL, DE
D06A	C1	POP BC
D06B	10 E9	DJNZ D056H
D06D	C3 00 D0	JP D000H

D080	00 00 00 E4 00 E5 00 00 00	タンクのデータ
D089	00 E4 87 87 87 87 87 E5 00	
D092	00 E6 EC EC 87 EC EC E7 00	

弾を発射する

タンクのデータ



UFOを画面の左から右に飛ばす

CP命令で比較、JP NZ命令で判定して繰り返しを繰り返す

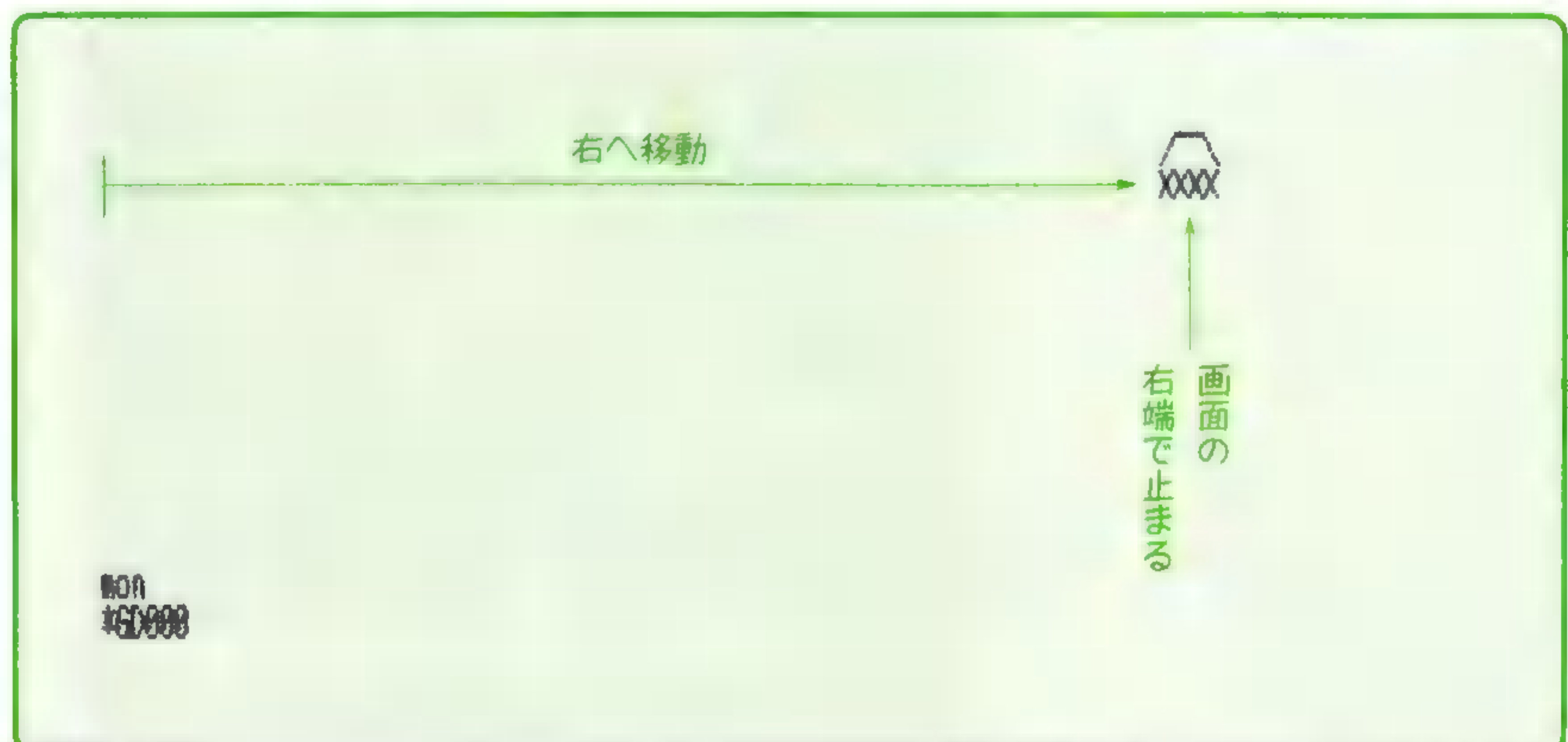
193頁でタンクというキャラクタの作り方を説明しました。また195頁では、タンクの表示の仕方を説明しましたが、タンクは固定したままでした。そこでここでは、UFOというキャラクタを取りあげて、キャラクタを移動させる方法について説明することにします。

まずUFOを、画面の左端から右端に向けて移動させます。この場合のポイントは、UFOが画面の右端にきたら移動を止めるということです。そのようにするには、CP命令とJP NZ命令を使って、UFOを移動させます。

CP命令は、CP命令のあとに置かれた数と、Aレジスタの内容とを比較する命令です。Aレジスタの内容と、CP命令に置かれた数とが等しくなると、ゼロ・フラグに1を立てます。Aレジスタの内容がCP命令に置かれた数より小さいうちは、ゼロ・フラグは0です（179～180頁参照）。

JP NZ命令は、ゼロ・フラグが0であるとき条件が成立して、JP命令に示されているメモリ番地にジャンプする命令です（179～180頁参照）。

では、このふたつの命令でどのようにすれば、つぎの図に示すように、UFOが画面の右端にきたら止まるようになるかです。



UFOを画面の左から右に飛ばす

このふたつの命令の働きをプログラムの説明のなかで行うとわかりにくくなるので、つぎに、UFOを画面の右に移動させるプログラムのなかから、その部分の命令を取り出して、おおまかに説明しておくことにします。アセンブリ言語の、それぞれの命令につけられているメモリ番地は、完成したプログラムに割り当てられるメモリ番地です。

```
D000    B, 01H
D005    PUSH BC
{
```

繰り返えす命令

UFOのデータを読み込む

UFOを表示していくメモリ番地の指定

```
{
D036    POP BC
D037    INC B
D039    LD A, B
D03A    CP 48H
D03C    JP NZ D005H
```

このプログラムのUFOが画面の右端にたどりつくには、UFOの表示を47H（10進数で71）回繰り返えして、画面の右へ移動させる必要があります。

この場合、CP命令で何回繰り返えしたか、繰り返えしの回数を比較させるので、まず、LD命令で繰り返えし回数の最初の数01H（10進数で1）を、Bレジスタに代入します。そして、PUSH BC命令で、Bレジスタの01Hをスタックのなかに退避させます。このようにするのは、Bレジスタをほかでも使うからです。レジスタは変数とおなじですから、あとから値が代入されると、まえに代入した値が破壊されてしまいます。それではカウントすることができなくなるので、スタックに退避させるわけです。

それから、その間にあるUFOを右へ表示していく命令を実行して、POP BC命令で、スタックに退避させた01Hを取り出してきました。

そして、INC B命令で、つぎに示すように、Bレジスタの01Hにプラス1します。INC命令は、プラス1する命令です。

01H+01H=02H

そこで、Bレジスタの内容は02Hになります。このBレジスタの02Hを、CP命令でCP命令に置かれた数と比較させます。ただし、CP命令はBレジスタとは比較できません。CP命令が比較する相手は、Aレジスタと決まっています。そのためBレジスタの02Hを、LD命令でAレジスタに代入してから、CP命令で、CP命令に置かれている数と、Aレジスタの内容とを比較させます。

UFOが画面の右端までたどりつくには、47H（10進数で71）回繰り返えせばよいわけですから、CP命令のあとに47Hを置いて、Aレジスタの内容が47Hであるかどうか比較させればよいということになりますが、ただこの場合は、最初にBレジスタに01Hを代入して、01Hから始めているので、

$$01H + 47H = 48H$$

CP命令に48H（10進数で72）を置いて、Aレジスタの内容が48Hになったかどうか比較させます。

Aレジスタの内容は02Hですから、CP命令にある48Hと等しくありません。そのため、ゼロ・フラグは0です。したがって、JP NZ命令の条件は成立して、指定されているメモリ番地D005Hにジャンプします。

メモリ番地のD005Hの命令は、PUSH BCですので、Bレジスタの02Hをスタックに退避します。理由はさきの場合とおなじです。

それから、その間にあるUFOを右へ表示していく命令を実行して、ふたたび、POP BC命令でスタックに退避した02Hを取り出してきました。そして、INC B命令でBレジスタの02Hに1をプラスします。

$$02H + 01H = 03H$$

Bレジスタの03Hを、LD命令でAレジスタに代入して、CP命令で比較します。Aレジスタの内容は03Hですから、まだ48Hとは等しくありません。そこで、JP NZ命令で、メモリ番地D005Hにジャンプします。

このことを47H回繰り返えしていくと、INC B命令で1ずつプラスされていって、Bレジスタの内容は48Hになります。Bレジスタの内容は、Aレジスタに代入されますから、Aレジスタの内容も、48Hになります。

Aレジスタの内容が48Hになると、CP命令に置かれている数と等しくなります。すると、CP命令の働きで、ゼロ・フラグに1が立ちます。

ゼロ・フラグに1が立つと、JP NZ命令の条件は成立しません。したがって、つぎのメモリ番地の命令の実行に移ります。

UFOを画面の左から右に飛ばす

ということは、メモリ番地D005番地にジャンプして、PUSH BC命令でBレジスタの内容をスタックに退避させたあと、実行してきたUFOを右へ表示する命令の実行も行わなくなります。そこでUFOの移動も止まるということになるのです。

UFOを画面の右へ向けて移動する

UFOを画面の右端へ移動するには、195頁でタンクを表示した場合と同様に、UFOのデータを読み取ります。ただこの場合は、タンクの表示の場合と違って、UFOを画面の右へひとつ進めるたびに、UFOのデータを読み取って表示しなければなりませんから、UFOのデータを読み取る命令も、繰り返えす命令のなかに入ります。したがって、順を追って説明することにします。

まず、カウン트의最初の数01Hを、LD命令でBレジスタに代入します。

```
D000    LD  B, 01H
```

つぎに、UFOを画面に表示する最初のメモリ番地を、HLレジスタに代入します。この場合は、メモリ番地F37AHからUFOを表示しますから、F37AHをHLレジスタに代入します。この表示位置はあとで示します。

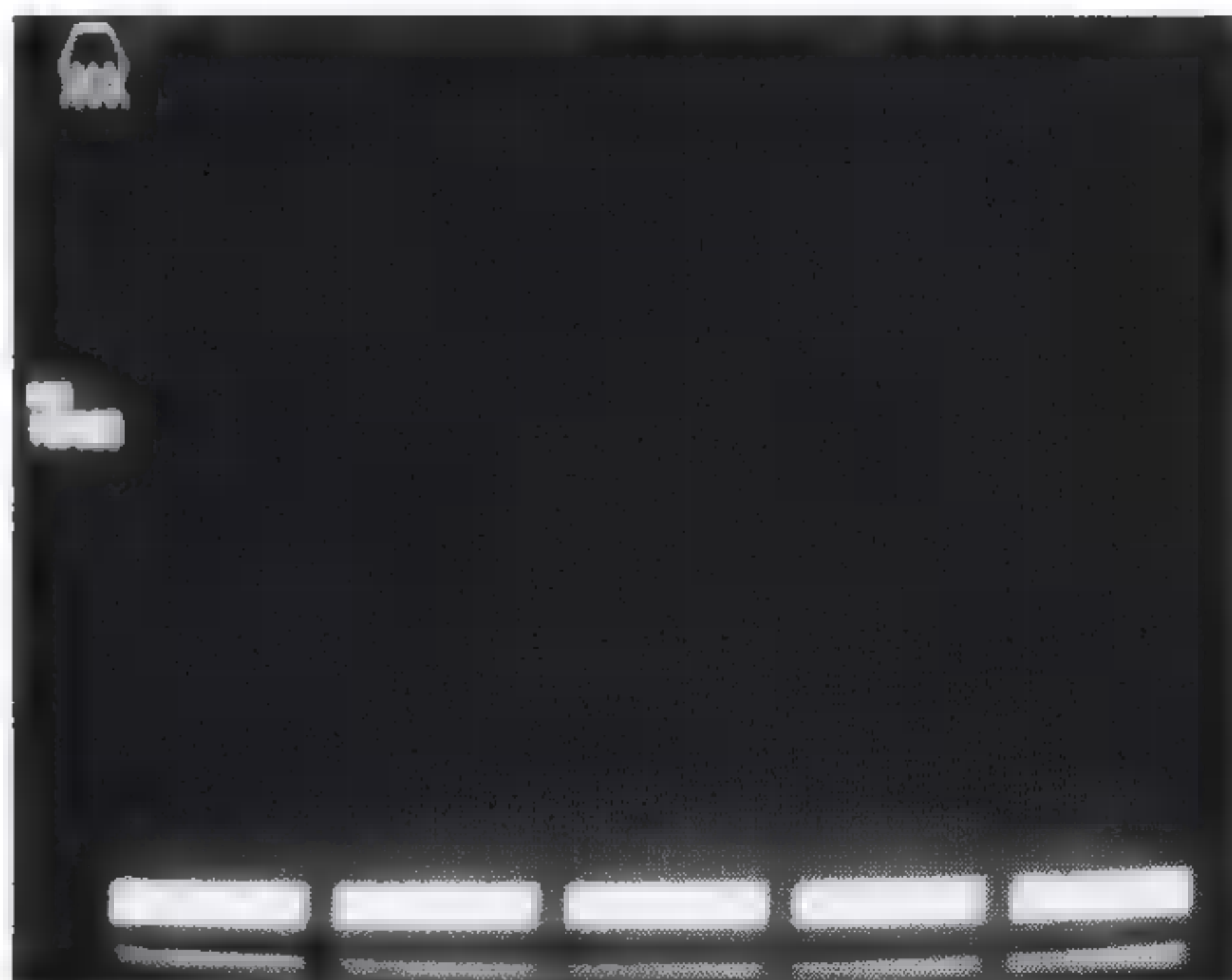
```
D002    LD  HL, F37AH
```

このあとで、Bレジスタ、HLレジスタともに、ほかのことに使いますから、代入した値を破壊しないために、PUSH命令でスタックに退避させます。

```
D005    PUSH BC
```

```
D006    PUSH HL
```

PUSH命令は16ビットの命令なので、PUSH命令のあとにBレジスタとか、Dレジスタだけを置くことはできません。このように、必ずレジスタペアにして置きます。



UFOのデータとデータを読み取る回数などの設定

さて、画面に表示するデータの読み取りですが、そのまえにUFOのデータを示します。データはつぎに示すように、メモリ番地D050Hから記憶します。このデータがキャラクタコードであることは193頁で説明しました。それぞれのキャラクタコードのキャラクタを、データの右に示しておきます。□は00H（空白）を示します。

D050	00	EE	94	□	／	—
D053	94	EF	00	—	＼	□
D056	00	F0	F0	□	×	×
D059	F0	F0	00	×	×	□

このデータは、つぎに示すように、画面に2行で表示しますから、繰り返しの命令DJNZ命令で、2回データを読み取る命令を繰り返せばよいことになります。そこで、LD命令でBレジスタに02H（10進数で2）を代入します。最初のLD B, 01Hの場合は、Bレジスタでなくてもかまいません。ただしこの場合は、DJNZ命令で繰り返えす回数をセットするわけですから、Bレジスタでなければなりません。

D007 LD B, 02H

そして、データを読み取るために、LD命令で、DEレジスタにデータが記憶されている最初のメモリ番地D050Hを代入します。

D009 LD DE, D050H

DEレジスタにデータが記憶されているメモリ番地を代入したら、ふたたびPUSH命令で、Bレジスタに代入した02Hを退避させます。なぜこのようにするかというと、データを2行目に表示するために、メモリ番地の計算をしなければならないからです。つまり、その場合にBレジスタを、BCレジスタペアにして使うので、Bレジスタの内容を退避させるわけです。

D00C PUSH BC

そして、もういちど、PUSH命令でHLレジスタのF37AHを、スタックに退避させます。

D00D PUSH HL

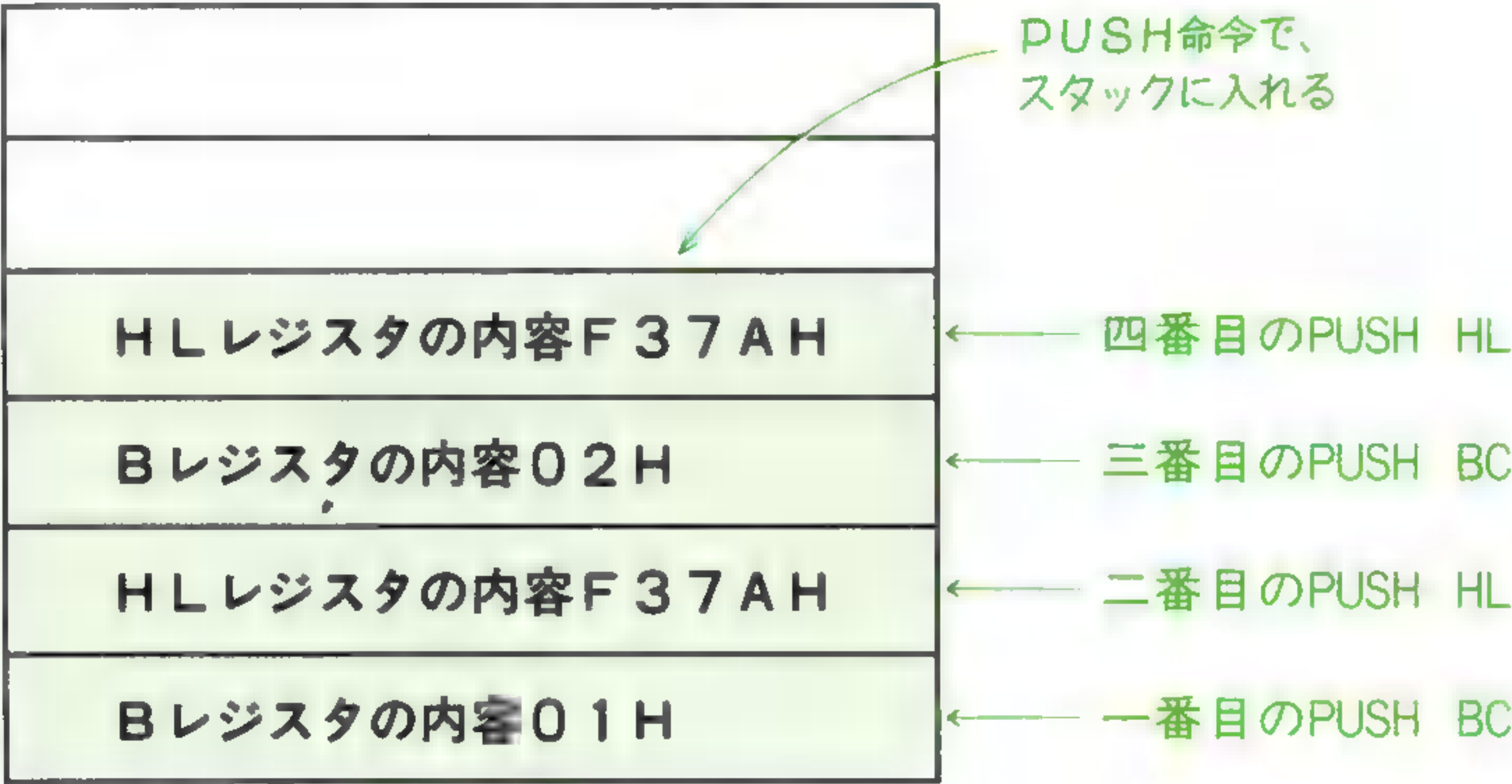
なぜ2回つづけて、HLレジスタのおなじ内容をスタックに退避させるのかは、つぎのスタックへの記憶のされ方で説明します。

スタックへのデータの記憶のされ方

ここまでの説明で、PUSH命令が4回でてきて、それぞれのデータをスタックに退避させました。ここで、データがスタックに、どのように記憶されているかみてみることにします。

スタックは、データを記憶する箱と考えてよいでしょう。スタックには、つぎの図に示すように、一番最初に入れたデータが一番下に、一番最後に入れたデータが一番上に、というように積み重なって記憶されます。

つまり、一番目のPUSH B命令で退避させた、Bレジスタの内容01Hが一番下に記憶されます。その上に、二番目のPUSH HL命令で、スタックに退避させたHLレジスタの内容F37AHが記憶され、その上に、三番目のPUSH BC命令で、スタックに退避させたBレジスタの内容02Hが記憶されます。一番上に、四番目のPUSH HL命令で、スタックに退避させたHLレジスタの内容F37AHが記憶されます。



スタック

PUSH命令で、スタックに記憶させたレジスタの内容を取り出す命令は、POP命令です。POP命令で、スタックからレジスタに内容を取り出すときは、スタックの一番上にあるデータから順に取り出してきます。下から二番目の内容が欲しいとしても、直接下から二番目の内容を取り出すことはできません。一番上にあるデータから順番に取り出してきて、そして、二番目のデータを取り出します。このようなデータの取り出し方を、ファースト・イン・ラスト・アウトと呼んでいます。

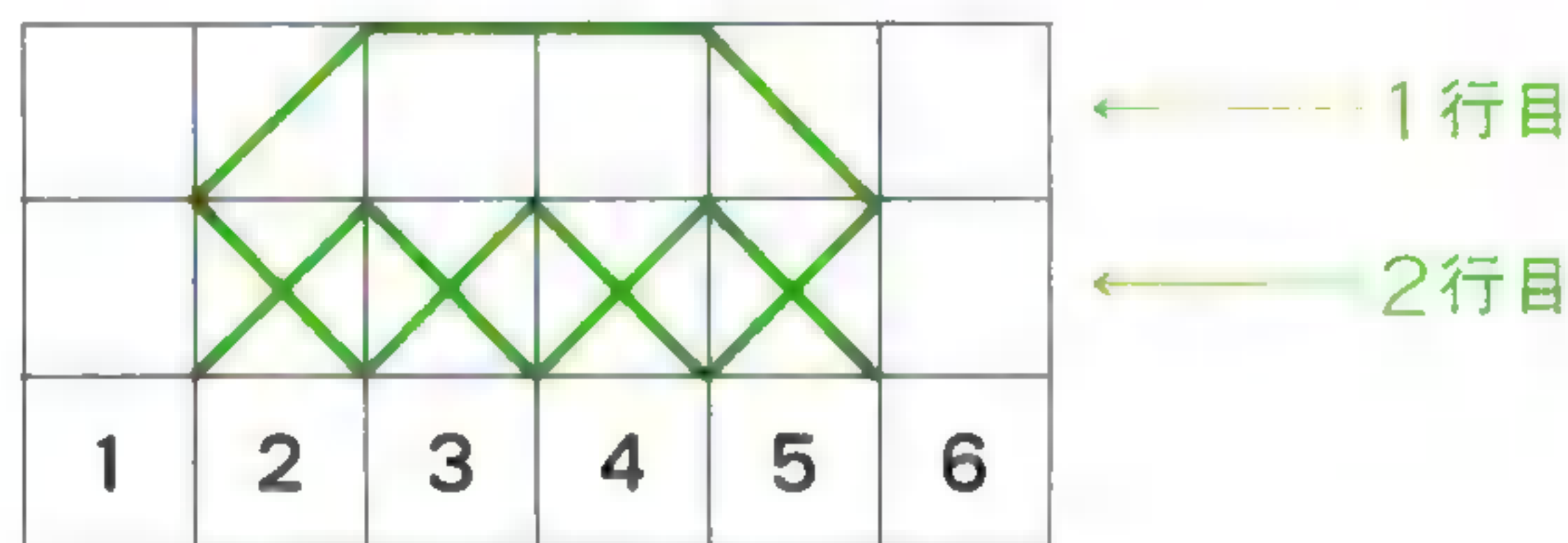
さて、四番目にPUSH HL命令で、二番目のPUSH HL命令とおなじよう

に、HLレジスタの内容F37AHをスタックに退避させましたが、四番目でおなじように、HLレジスタの内容をスタックに退避させておかないと、POP HL命令で、HLレジスタに取り出されてくるデータが三番目のBレジスタの内容になってしまいます。したがって、まえに説明したように、2行目のキャラクタが表示されなくなってしまうからです。

もちろん、POP BC命令がPOP HL命令により先に実行されて、三番目のデータが先に取り出されるようなプログラムであれば、四番目のPUSH HL命令は必要ないことになります。

UFOのデータを読み取って表示

UFOは、2行で表示します。UFOの1行分は、この場合、つぎの図に示すように6個です。



UFOの表示も、195頁で説明したタンクの表示の場合とおなじです。UFOの1行目の最初のキャラクタを表示するVRAMのメモリ番地は、

D002 LD HL, F37AH

で指定しています。したがって、INC HL命令で、HLレジスタの内容に1をプラスしていったVRAMのメモリ番地を指定して、1行目のキャラクタ6個を表示します。

さて、タンクの表示の場合、2行目のキャラクタを表示するにはどうしたでしょうか。1行目のキャラクタが6個全部表示されたかどうかを判定させて、つぎにHLレジスタの内容に、0078H（10進数で120）をプラスして、2行目の最初のキャラクタを表示するVRAMのメモリ番地を指定して、表示しました。キャラクタを6個表示したかどうかを判定させるには、カウントする必要があります。そこで、カウントの最初の値を、LD命令でCレジスタに代入します。

D00E LD C, 01H

つぎは、データの読み取りです。データは、メモリ番地D050Hから記憶

されています。そして、データが最初に記憶されているメモリ番地は、DEレジスタに代入してあります。したがって、DEレジスタに記憶されているメモリ番地が指定する内容を、Aレジスタに代入します。この場合、

LD A, DE ← ~~間違い~~

とすると、間違いです。このようにすると、DEレジスタに記憶されているメモリ番地F37AH、そのものが代入されてしまいます。

メモリ番地F37AHに記憶されているものをAレジスタに代入するには、DEレジスタをカッコで囲んで、

LD A, (DE) ← ~~正しい~~

とします。このことをよくおぼえておかなければなりません。

D010 LD A, (DE) ← DEレジスタで指定した内容が
Aレジスタに代入される

つぎは、Aレジスタに代入されたキャラクタの表示です。キャラクタを表示するVRAMのメモリ番地は、HLレジスタに記憶されていますから、Aレジスタの内容を、LD命令でHLレジスタに代入します。

D011 LD (HL), A

これで、VRAMのメモリ番地F37AHに、最初のキャラクタが表示されます。ただし、最初のキャラクタは、00Hですから空白です。したがって何も表示されません。この空白は195頁のタンクの表示のところで説明しましたが、UFOを左右に移動するときに、前に表示したUFOの最後の1個を消すためのものです。

つぎのキャラクタの表示のために、INC命令で1プラス

1個のキャラクタを表示したら、つぎのキャラクタを表示するために、INC命令で、HLレジスタの内容に1をプラスします。

D012 INC HL

このようにすると、HLレジスタの内容は、

F37AH+01H=F37BH

で、F37BHとなります。つまり、VRAMのメモリ番地F37AHの、つぎのメモリ番地となります。まだやることがあります。つぎに読み取るデータのメモリ番地の指定です。メモリ番地を指定しないと、つぎのデータの読み取りができないからです。データのメモリ番地は、DEレジスタが記憶していますから、DEレジスタの内容に、INC命令で1をプラスします。

D013 INC DE

このようにすると、DEレジスタの内容は、

D050H+01H=D051H

で、D051H、つまり、つぎのデータが記憶されているメモリ番地となります。そして、カウントするために、Cレジスタの内容にINC命令で、プラス1します。

D014 INC C

6回繰り返えしたが比較する

INC命令でCレジスタの内容に1をプラスしたら、比較する命令CP命令で、Cレジスタの内容が07H（10進数で7）になったかどうか比較させるために、LD命令で、AレジスタにCレジスタの内容を代入します。

D015 LD A, C

比較の命令CP命令のあとに、カウントする回数、つまり繰り返えす回数07Hを置きます。

D016 CP 07H

CP命令のあとの値が、06Hではなくて07Hになっているのは、カウントする最初の値として、Cレジスタに01Hを代入しているからです。

さてCP命令は、07HとAレジスタの内容を比較します。07HとAレジスタの内容が等しいと、CP命令の働きによって、ゼロ・フラグに1が立ちます。まだ、INC C命令は1回しか実行されていませんから、当然、Aレジスタは07Hになっていません。したがって、ゼロ・フラグは0です。

ゼロ・フラグが0のときは、まだ1行目のキャラクタが全部表示されていませんから、キャラクタを読み込むLD A, (DE)命令に戻します。

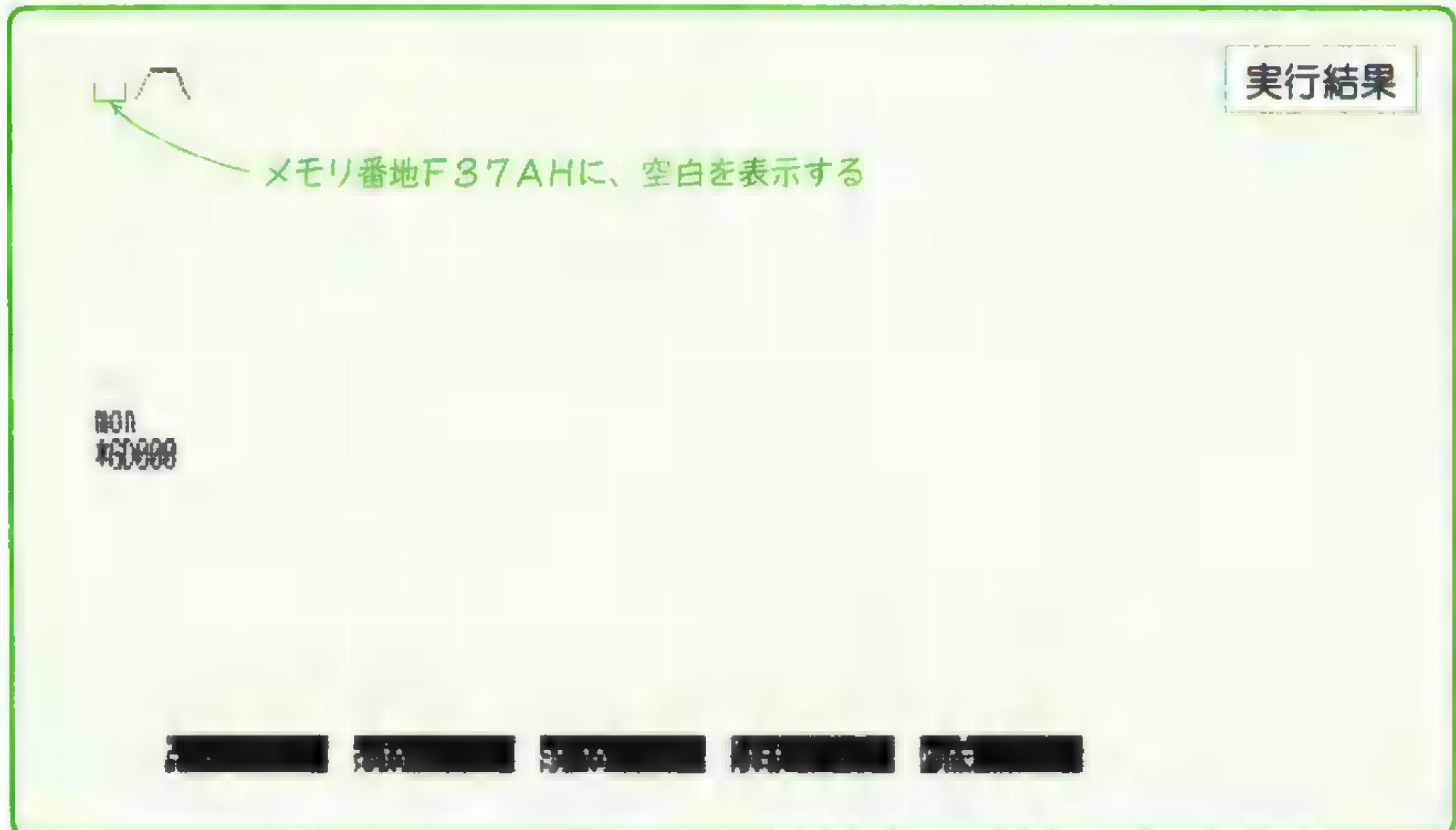
ゼロ・フラグが0のとき、条件が成立して指定されたメモリ番地にジャンプする命令は、JP NZ命令です。したがって、JP NZ命令にD010Hを置きます。

D018 JP NZ D010H

これで、空白を表示したVRAMのメモリ番地F37AHの、つぎのVRAMのメモリ番地F37BHに、2個目のキャラクタが表示されます。

このようにして、Aレジスタの内容が、CP命令に置かれている07Hになるまで、繰り返えされていきます。Aレジスタの内容が、CP命令の比較によ

って07Hと等しいとわかると、ゼロ・フラグに1が立ちます。ゼロ・フラグに1が立つと、JP NZ命令の条件は成立しませんから、ジャンプは終わって、つぎの命令の実行に移ります。つまりここまでで、つぎの実行結果に示すように、1行目のキャラクタ6個全部が表示されます。



2行目のキャラクタの表示

CP命令とJP NZ命令によって1行目のキャラクタを表示すると、つぎは、2行目のキャラクタの表示を行います。2行目のキャラクタを表示するには、VRAMのメモリ番地F37AHに、0078H（10進数で120）をプラスします。120をプラスする理由は150頁を参照してください。

HLレジスタに記憶されているVRAMのメモリ番地は、INC HL命令で変化してしまっていますから、四番目のPUSH命令で、スタックに退避させたHLレジスタの内容F37AHを、POP命令で取り出します。

D01B POP HL

スタックから取り出してきたF37AHに、0078Hを足すには、まず0078Hを、LD命令でレジスタペアに代入します。ここでは、BCレジスタを使います。

D01C LD BC, 0078H

そして、加算の命令であるADD命令を使って、HLとBCレジスタの内容を足します。

D01F ADD HL, BC

HLレジスタの内容とBCレジスタの内容を足すと、

F37AH+0078H=F3F2H

で、F3F2Hとなります。このF3F2Hは、HLレジスタに記憶されます。F3F2Hは、F37AHの真下のメモリ番地です。

2行目の最初のキャラクタの表示位置を算出したら、データを読み取るために、繰り返えしの命令DJNZ命令で、繰り返えし処理をする最初の命令に戻しますが、そのまえに、三番目にPUSH BC命令でスタックに退避させたBレジスタの内容02Hを、POP BC命令でBレジスタに取り出します。

D020 POP BC

このようにするのは、DJNZ命令は、Bレジスタの内容02Hから1を引いて、繰り返えし処理をする最初の命令に戻るからです。

D021 DJNZ D00CH

DJNZ命令のあとにあるD00CHが、繰り返えし処理をする命令が記憶されているメモリ番地です。メモリ番地D00CHに記憶されている命令は、PUSH BC命令です。そこでDJNZ命令で、1を引いた結果のBレジスタの内容01Hを、スタックに退避します。つぎにPUSH HLで、HLレジスタの内容F3F2Hもスタックに退避します。

そして、ふたたびさきに説明したように、LD A, (DE)命令とLD (HL), A命令、INC HL命令とINC DE命令などを繰り返えして、2行目のキャラクタを読み取って表示していきます。

キャラクタを何個表示したかは、INC C命令でカウントしていますから、Cレジスタの値が07Hになると、Aレジスタの内容も07Hになります。したがって、比較する命令CP命令で07Hと比較されて、ゼロ・フラグに1が立ちます。ゼロ・フラグが1になると、JP NZ命令の条件は成立しませんから、つぎのメモリ番地の命令の実行に移ります。

メモリ番地D01BHのPOP HL命令からD01FHのADD HL, BC命令までを、まえに説明したように行って、POP BC命令でスタックに退避させた01HをBレジスタに取り出してきます。そして、DJNZ命令でBレジスタの内容から1を引きます。

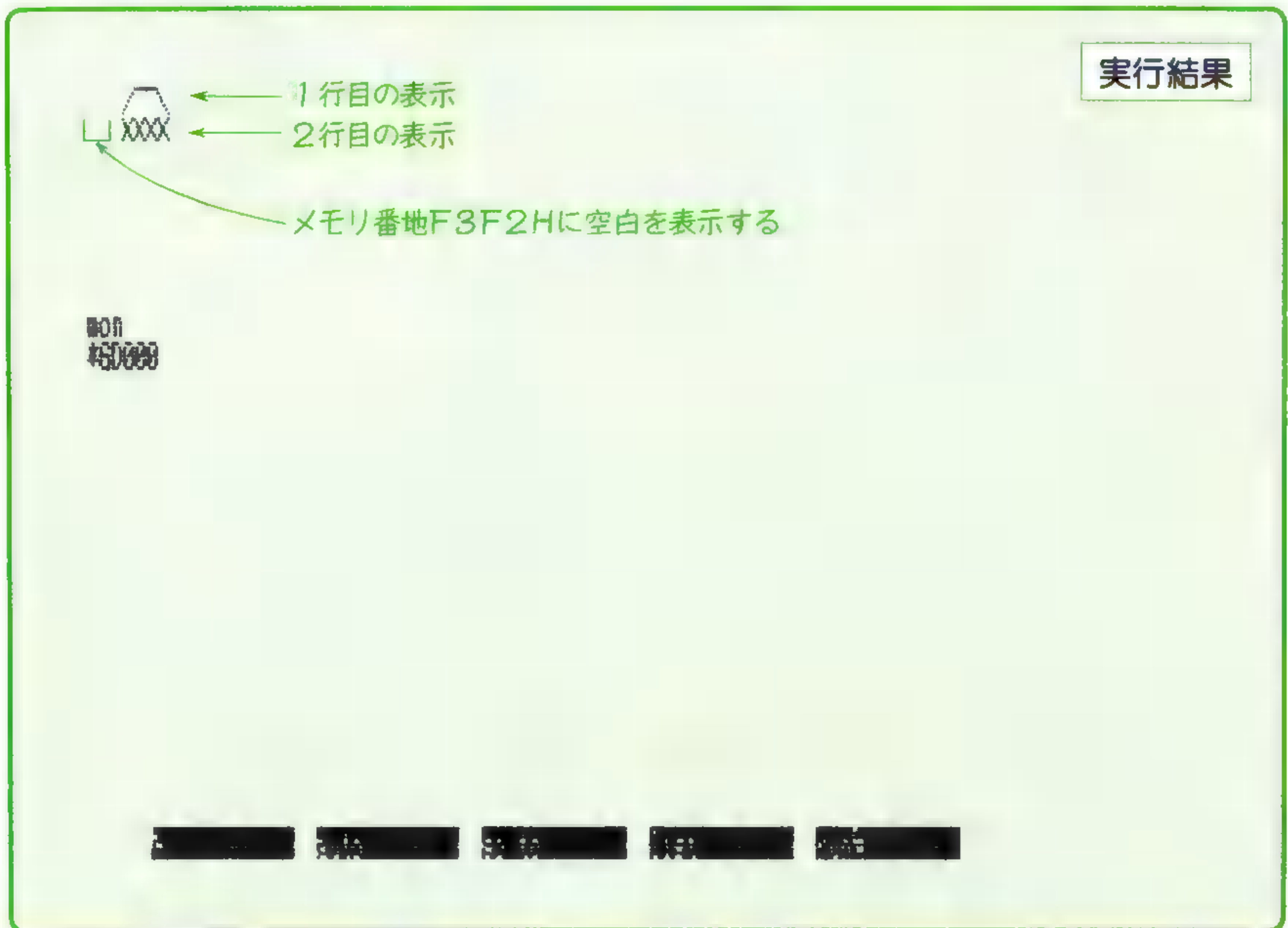
01H-01H=00H

1を引くと、Bレジスタは0になりますから、DJNZ命令による繰り返えし

UFOの画面の左から右に飛ばす

は終了して、つぎのメモリ番地の命令の実行に移ります。

つぎに、2回目の命令を実行して、2行目のキャラクタを表示した実行結果を示します。

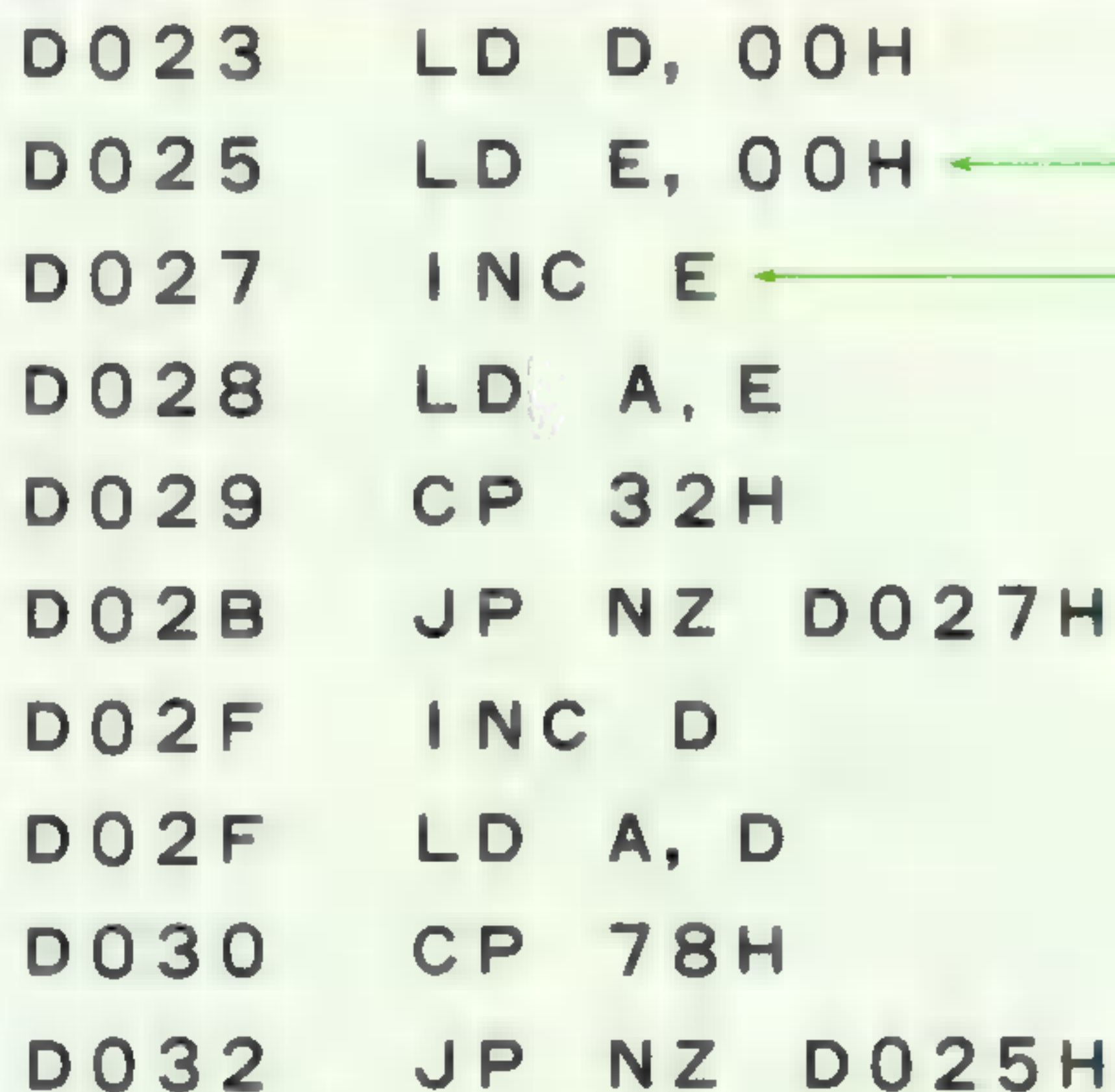


ウェイト・ループを入れる

2行目のキャラクタを表示して、すぐにつぎの命令を実行させてしまうと、高速で命令が処理されるので、キャラクタの左側にある空白でUFOが消されていってしまい、UFOの移動をみることができません。

そこで、つぎの命令の実行を送らせるために、ここにウェイト・ループを入れます。ウェイト・ループについては178頁で説明していますので、その頁を参照してください。ウェイト・ループは、そっくりそのまま、ここに挿入したにすぎないので、説明は省略します。

```
D023    LD  D, 00H
D025    LD  E, 00H
D027    INC E
D028    LD  A, E
D029    CP  32H
D02B    JP  NZ D027H
D02F    INC D
D02F    LD  A, D
D030    CP  78H
D032    JP  NZ D025H
```



UFOを表示するつぎのメモリ番地の指定

UFOを1個表示して、ウェイト・ループで、つぎの命令の実行を遅らせました。つぎは、つぎに表示するUFOのメモリ番地の指定を行います。それとともに、何回UFOを繰り返えして表示したかの判定も行います。この判定は、最初に説明したように、UFOが画面の右端にきたとき、そこでUFOの移動を止めるためです。

まず、つぎにUFOを表示するVRAMのメモリ番地を指定するために、二番目のPUSH HL命令で、スタックに退避させたHLレジスタの内容F37AHを、POP命令でHLレジスタに戻します。

```
D035    POP HL
```

POP HL命令でHLレジスタに取り出されてきたF37AHは、UFOの最

UFOを画面の左から右に飛ばす

初のキャラクタを、画面の左端に表示したVRAMのメモリ番地です。UFOを画面の右にひとつ移動させるために、HLレジスタの内容F37AHに1プラスします。1プラスする命令はINC命令です。

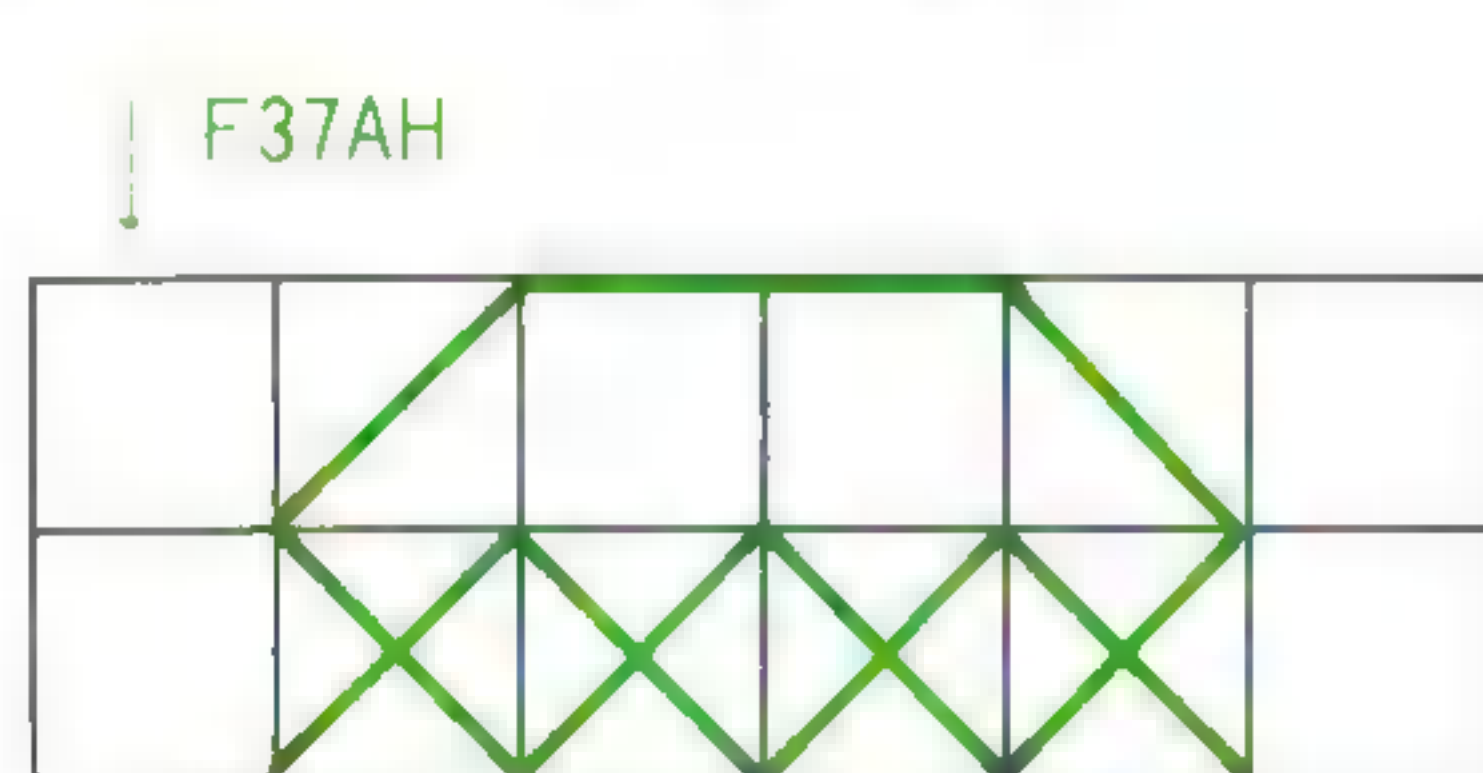
D038 INC HL

POP HL命令とINC HL命令の間にふたつの命令が入りますが、説明の都合上、あとにまわします。

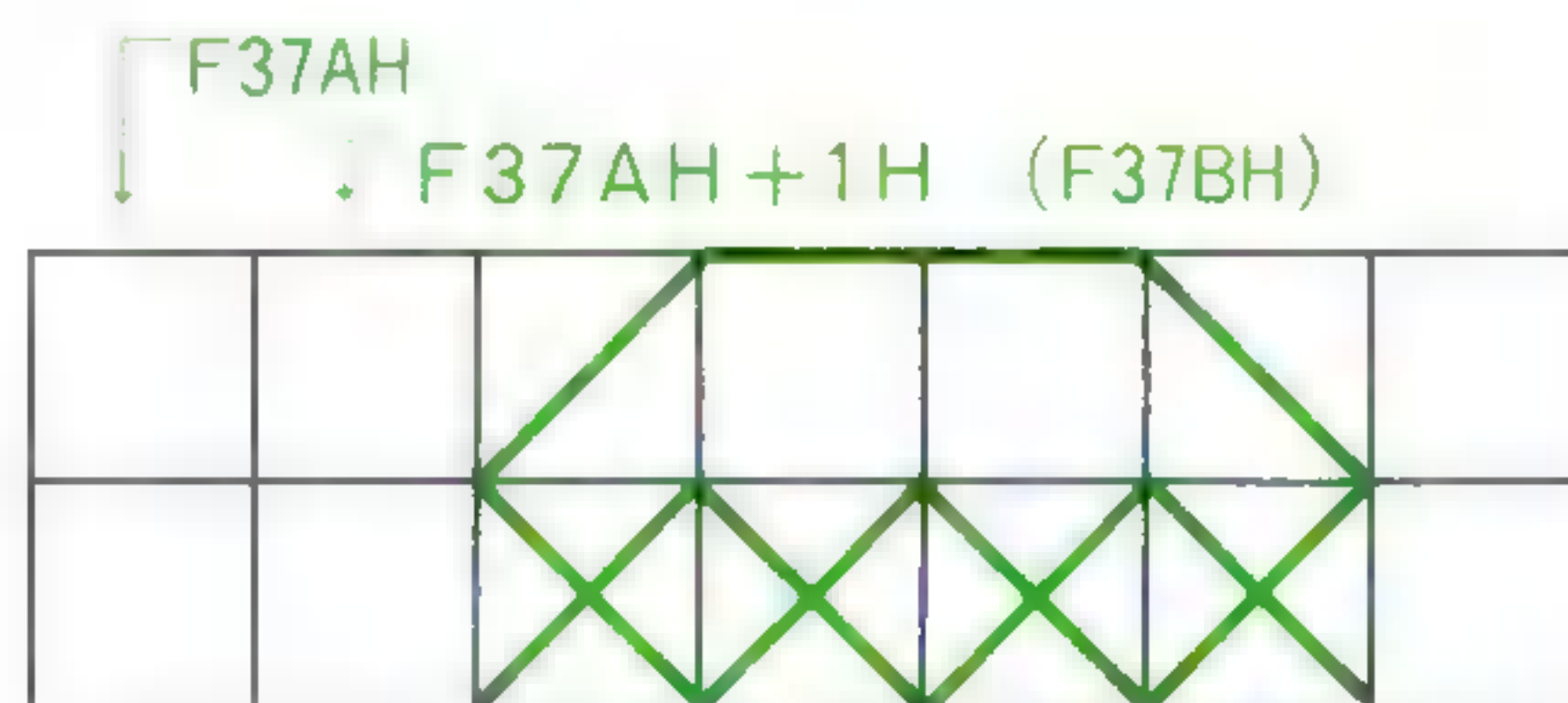
INC HL命令で、HLレジスタの内容に1をプラスすると、

$$F37AH + 01H = F37BH$$

F37BHがHLレジスタの内容となります。したがって、このHLレジスタの内容を、JP NZ命令で、まえに説明したUFOのデータを読み取る命令に戻してやると、つぎの図に示すVRAMのメモリ番地から、つぎのUFOが表示されていくことになります。



1 個目のUFOの表示



2 個目のUFOの表示

上の図をみるとわかるように、つぎのUFOの表示は、まえのUFOの上にかぶさって表示されますから、まえのUFOは消されることになるわけです。左側に空白がないと、UFOの最後のキャラクタが残ってしまうこともわかっていきます。

48回繰り返し運えたかどうかの判定

では、あとに説明をまわした命令です。一番目のPUSH BC命令で、画面の右に向かって47H（10進数で71）回、UFOを表示していくために、カウントする最初の数01Hをスタックに退避させました。その01Hを、POP命令でBレジスタに取り出します。

D036 POP BC

POP BC命令でBレジスタに取り出した内容に、INC命令で1プラスします。

D037 INC B

Bレジスタの内容が01Hのときは、

01H+01H=02H

Bレジスタの内容は、02Hとなります。つぎにBレジスタの内容が、47H（10進数で71）回になったかどうか、CP命令で比較させるためにLD命令で、Bレジスタの内容をAレジスタに代入します。

D039 LD A, B

そして、CP命令で比較させます。CP命令のあとに置く数は48Hです。

D03A CP 48H

CP命令で、Aレジスタの内容と48Hとを比較した結果、Aレジスタの内容と48Hとが等しいとき、ゼロ・フラグに1が立ちます。48Hにならないうちは、ゼロ・フラグは0です。ゼロ・フラグが0のとき、条件が成立して、指定されたメモリ番地にジャンプする命令は、JP NZ命令です。

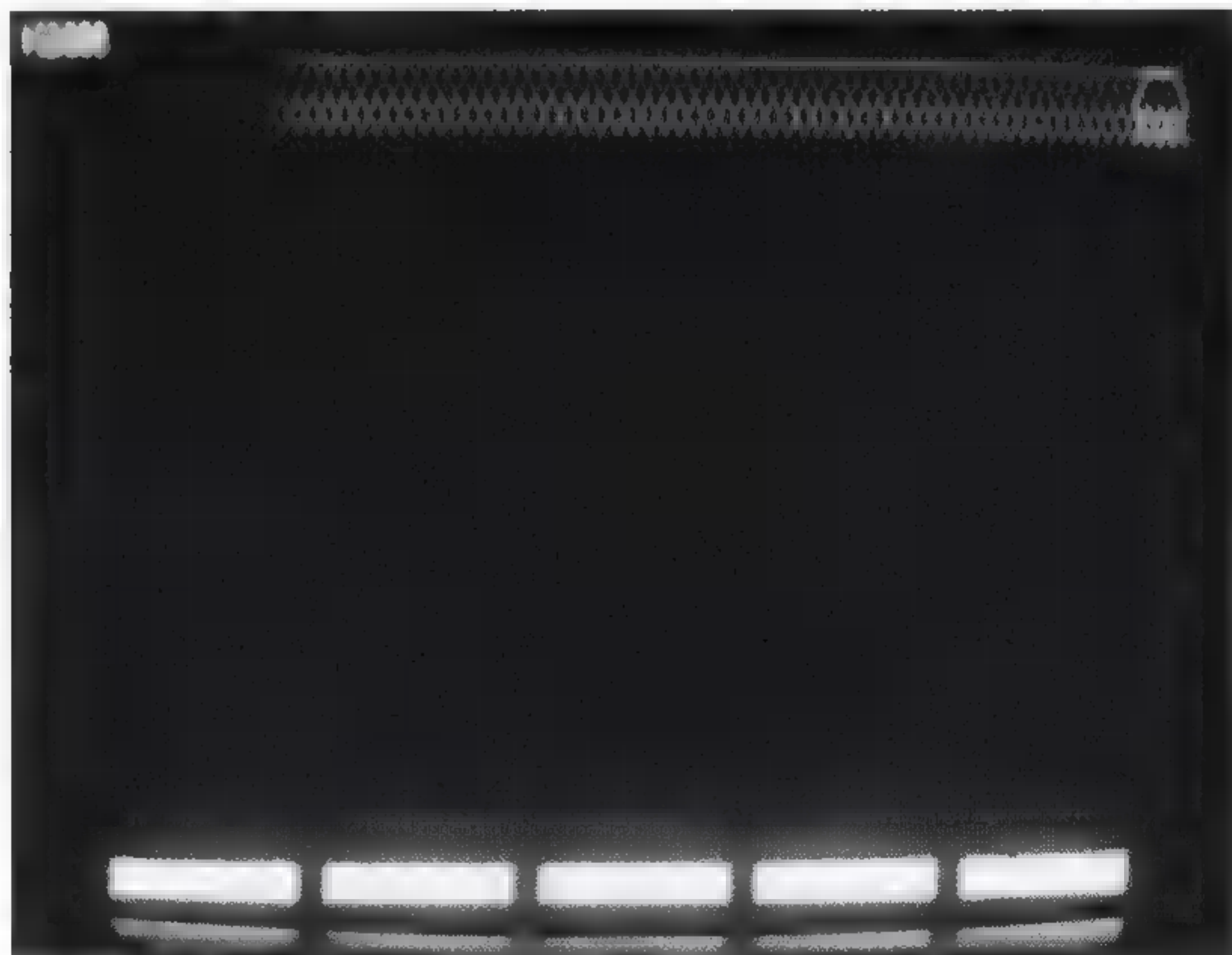
D03C JP NZ D005H

さきに説明したように、INC B命令でBレジスタの内容に1プラスしていった、Bレジスタの内容が48Hに等しくなるまで、JP NZ命令で、メモリ番地D005Hに戻り、画面の左から右へUFOを表示していきます。

47H回繰り返えすと、Aレジスタの内容、つまりBレジスタの内容が48Hになりますから、CP命令で48Hと比較されて、ゼロ・フラグに1が立ちます。したがって、JP NZ命令の条件が成立しなくなりますから、ジャンプはやめて、つぎの命令の実行に移ります。

D03F JP 5C66H

では、つぎに実行結果を示します。



アセンブルしたプログラム

ではつぎに、アセンブルしたプログラムを示します。このプログラムを実行するときは、画面を80×25モードにして実行してください。

メモリ番地	マシン語	アセンブリ言語
D000	06 01	LD B, 01H
D002	21 7A F3	LD HL, F37AH
D005	C5	PUSH BC
D006	E5	PUSH HL
D007	06 02	LD B, 02H
D009	11 50 D0	LD DE, D050H
D00C	C5	PUSH BC
D00D	E5	PUSH HL
D00E	0E 01	LD C, 01H
D010	1A	LD A, (DE)
D011	77	LD (HL), A
D012	23	INC HL
D013	13	INC DE
D014	0C	INC C
D015	79	LD A, C
D016	FE 07	CP 07H
D018	C2 10 D0	JP NZ D010H
D01B	E1	POP HL
D01C	01 78 00	LD BC, 0078H
D01F	09	ADD HL, BC
D020	C1	POP BC
D021	10 E9	DJNZ D00CH
D023	16 00	LD D, 00H
D025	1E 00	LD E, 00H
D027	1C	INC E

D028	7B	LD A, E
D029	FE 32	CP 32H
D02B	C2 27 D0	JP NZ D027H
D02E	14	INC D
D02F	7A	LD A, D
D030	FE 78	CP 78H
D032	C2 25 D0	JP NZ D025H
D035	E1	POP HL
D036	C1	POP BC
D037	04	INC B
D038	23	INC HL
D039	78	LD A, B
D03A	FE 48	CP 48H
D03C	C2 05 D0	JP NZ D005H
D03F	C3 66 5C	JP 5C66H

UFOのデータ

D050	00 EE 94
D053	94 EF 00
D056	00 F0 F0
D059	F0 F0 00



UFOを画面の右から左へ飛ばす

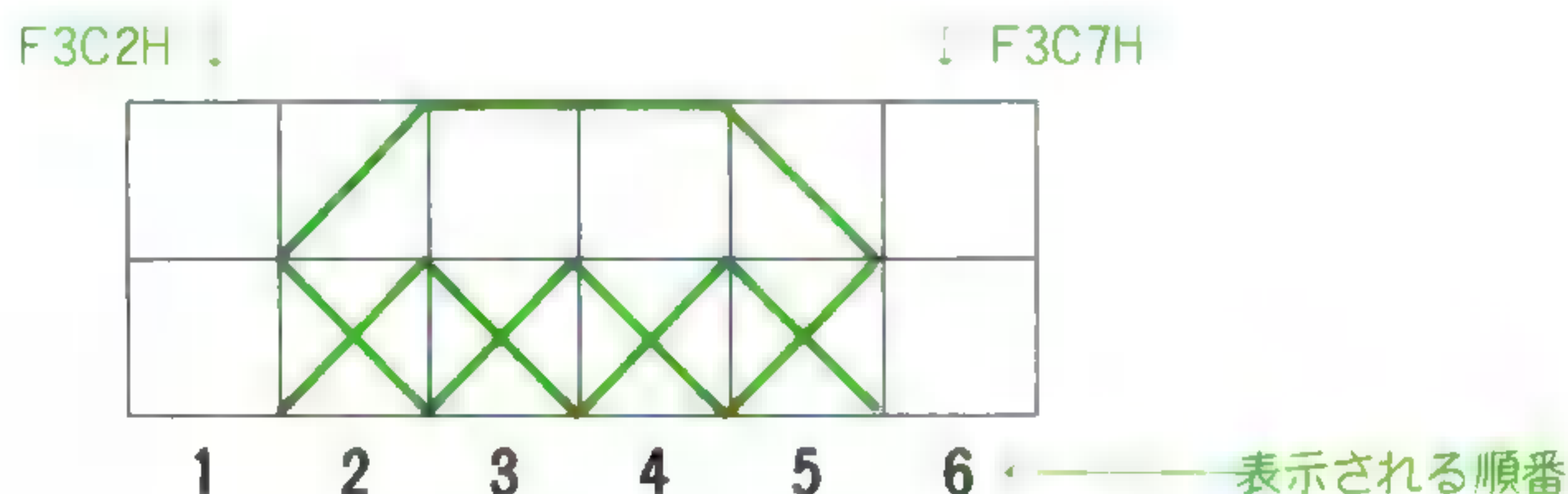
INC HL命令も、DEC HL命令に変える

216頁でUFOを取りあげて、キャラクタを画面の左から右へ移動させる方法について説明しました。キャラクタの移動は、画面の左から右というばかりではなく、画面の右から左という場合もあるでしょう。そこでここでは、UFOを、画面の右から左へ移動する方法について説明することにします。

UFOを画面の右から左へ移動する方法といっても、UFOを画面の左から右へ移動する方法と、ほとんど変わるところはありません。

変わるところは、つぎの2箇所だけです。

ひとつは、最初にUFOを表示するVRAMのメモリ番地を、画面の右端より6個左側のメモリ番地F3C2Hにすることです。すると、つぎの図に示すように、UFOが表示されます。



もうひとつは、UFOを画面の左から右へ移動する場合、メモリ番地D038Hの命令で、

D038 INC HL ← 1プラスする

として、INC HL命令で、VRAMのメモリ番地を記憶しているHLレジスタに1プラスしましたが、右から左へ移動させる場合は、この命令を、

D038 DEC HL ← 1マイナスする

に変えて、DEC命令で、HLレジスタの内容から1を引いていくということです。DEC命令については132頁を参照してください。

なぜ、DEC HL命令で、HLレジスタの内容から1を引くのかというと、画面の左から右へという場合は、VRAMのメモリ番地はひとつずつ増えていきま

すが、画面の右から左へという場合は、VRAMのメモリ番地は、ひとつずつ減っていくからです。

この二点が変わるだけで、あとはすべて命令はおなじですから、つぎにプログラムを示すにとどめます。なお、プログラムを実行するときは、画面を80×25モードにして実行してください。

アセンブルしたプログラム

メモリ番地	マシン語	アセンブリ言語
D000	06 01	LD B, 01H
D002	21 C2 F3	LD HL, F3C2H
D005	C5	PUSH BC
D006	E5	PUSH HL
D007	06 02	LD B, 02H
D009	11 50 D0	LD DE, D050H
D00C	C5	PUSH BC
D00D	E5	PUSH HL
D00E	0E 01	LD C, 01H
D010	1A	LD A, (DE)
D011	77	LD (HL), A
D012	23	INC HL
D013	13	INC DE
D014	0C	INC C
D015	79	LD A, C
D016	FE 07	CP 07H
D018	C2 10 D0	JP NZ D010H
D01B	E1	POP HL
D01C	01 78 00	LD BC, 0078H
D01F	09	ADD HL, BC
D020	C1	POP BC
D021	10 E9	DJNZ D00CH

UFOを画面の右から左へ飛ばす

D023	16	00		LD D, 00H
D025	1E	00		LD E, 00H
D027	1C			INC E
D028	7B			LD A, E
D029	FE	32		CP 32H
D02B	C2	27	D0	JP NZ D027H
D02E	14			INC D
D02F	7A			LD A, D
D030	FE	78		CP 78H
D032	C2	25	D0	JP NZ D025H
D035	E1			POP HL
D036	C1			POP BC
D037	04			INC B
D038	2B			DEC HL
D039	78			LD A, B
D03A	FE	48		CP 48H
D03C	C2	05	D0	JP NZ D005H
D03F	C3	66	5C	JP 5C66H

UFOのデータ

D050	00	EE	94
D053	94	EF	00
D056	00	F0	F0
D059	F0	F0	00

画面の左右にきたUFOの消去

00HをUFOの上に表示する

216頁で、UFOを画面の左から右へ移動させる方法、また233頁では、UFOを画面の右から左へ移動させる方法について説明しました。

この両方のプログラムを実行させると、画面の両端でUFOが表示されたままになっています。このまま表示しておいたのでは、なにかとジャマになります。そこでここでは、UFOが画面の右端にきたとき、UFOを消す方法について説明することにします。

画面の右端まで移動してきて、表示されたままになっているUFOを消去するには、UFOの大きさだけの空白を、UFOが表示されている位置の上に表示して消します。空白のキャラクタコードは、00Hです。

この場合、UFOは2行でできています。そして、1行目、2行目ともに6個のキャラクタでできていましたから、最初に、UFOが表示されている1行目のVRAMのメモリ番地を指定して、00Hを6個表示します。つぎに、2行目のVRAMのメモリ番地を指定して、00Hを6個表示します。このように00Hを表示して、UFOの消去を行うわけです。

まず、2行にわたって00Hを表示しますから、繰り返えす回数02H（10進数で2）を、LD命令でBレジスタに代入します。

LD B, 02H

この場合は、DJNZ命令で繰り返えすので、繰り返えす回数は、Bレジスタに代入しなければなりません。



00Hを表示する最初のVRAMのメモリ番地を求める

つぎに、UFOが表示されているVRAMのメモリ番地を、LD命令でHLレジスタに代入します。HLレジスタに代入するVRAMのメモリ番地は、つぎのようにして求めます。

$$\text{F37AH} + 46\text{H} \text{ (10進数で70) } = \text{F3C0H}$$

繰り返えす回数から1を引いた数

最初に表示するVRAMのメモリ番地

この場合、UFOの表示は、VRAMのメモリ番地F37AHから始まって71回繰り返えすと、UFOは画面の右端にきます。したがって、F37AHに繰り返えす回数から1を引いた数70を加えればよいのです。70は10進数ですから、これを16進数46Hになおして足します。

この足し算で求められたF3C0Hが、画面の右端に表示されているUFOの左側のVRAMのメモリ番地となります。つまり、00Hを表示する最初のVRAMのメモリ番地ということです。そこで、このVRAMのメモリ番地F3C0Hを、LD命令でHLレジスタに代入します。

LD HL, F3C0H

つぎに、2行目に00Hを表示するために、2行目の最初のVRAMのメモリ番地を算出する値を、LD命令でDEレジスタに代入します。

2行目の最初のVRAMのメモリ番地は、1行目の最初のVRAMのメモリ番地に、0078H（10進数で120）を足せばよいわけです（150頁参照）。したがって、つぎのように、DEレジスタに0078Hを代入します。

LD DE, 0078H

02Hを代入したBレジスタと、F3C0Hを代入したHLレジスタには、もう一度ほかの値を代入しますから、02HとF3C0Hが破壊されないために、PUSH命令でスタックに退避させます。

PUSH BC

PUSH HL

さて00Hを表示する、最初のVRAMのメモリ番地は決まりましたが、UFOは1行6個のキャラクタでできていますから、さきに説明したように、1行に00Hを6個表示させなければなりません。1行に6個の00Hを表示するためには、00Hの表示を6回繰り返えす必要があります。そこで、繰り返えす回数06H（10進数で6）を、LD命令でBレジスタに代入します。

LD B, 06H

これで00Hを表示する準備は整いましたから、つぎは00Hの表示です。

00Hの表示

00Hを画面に表示するには、LD命令で00HをAレジスタに代入します。

LD A, 00H

つぎに、Aレジスタの内容00Hを、LD命令で、VRAMのメモリ番地を記憶しているHLレジスタに代入します。

LD (HL), A

1回目の実行のとき、HLレジスタの内容は、VRAMのメモリ番地F3C0Hですから、00HはF3C0Hに表示されます。

LD (HL), A
F3C0H ← 代入 00H

00Hを1個表示して、キャラクタを1個消去したら、つぎのキャラクタの消去を行わなければなりません。そうするには、VRAMのメモリ番地に1をプラスして、つぎのVRAMのメモリ番地を求めて指定します。1プラスする命令は、INC命令です。1プラスする内容はHLレジスタに記憶されていますから、

INC HL

となります。INC HL命令が実行されると、

F3C0H + 01H = F3C1H

となって、F3C1HがHLレジスタの内容になります。したがって、このF3C1Hを、00Hを表示するLD (HL), A命令にDJNZ命令で送ってやると、VRAMのメモリ番地F3C1Hに00Hが表示されて、つぎのキャラクタが消去されます。

DJNZ <LD (HL), Aのメモリ番地>

DJNZ命令は、Bレジスタに代入した06H（10進数で6）から1を引いて、指定されたメモリ番地にジャンプします。Bレジスタに代入した06Hが0になると、DJNZ命令によるジャンプは終了して、つぎのメモリ番地の命令の実行に移りますから、1行目の6個のキャラクタの上に00Hを表示し終わるまで、LD (HL), A命令にジャンプすることになります。

2行目の最初のVRAMのメモリ番地を求める

このようにDJNZ命令でジャンプして、1行目のキャラクタの上に00Hを表示し終わると、つぎの命令の実行に移りますから、PUSH HL命令でスタックに退避させたVRAMのメモリ番地F3C0Hを、POP命令で取り出します。

POP HL

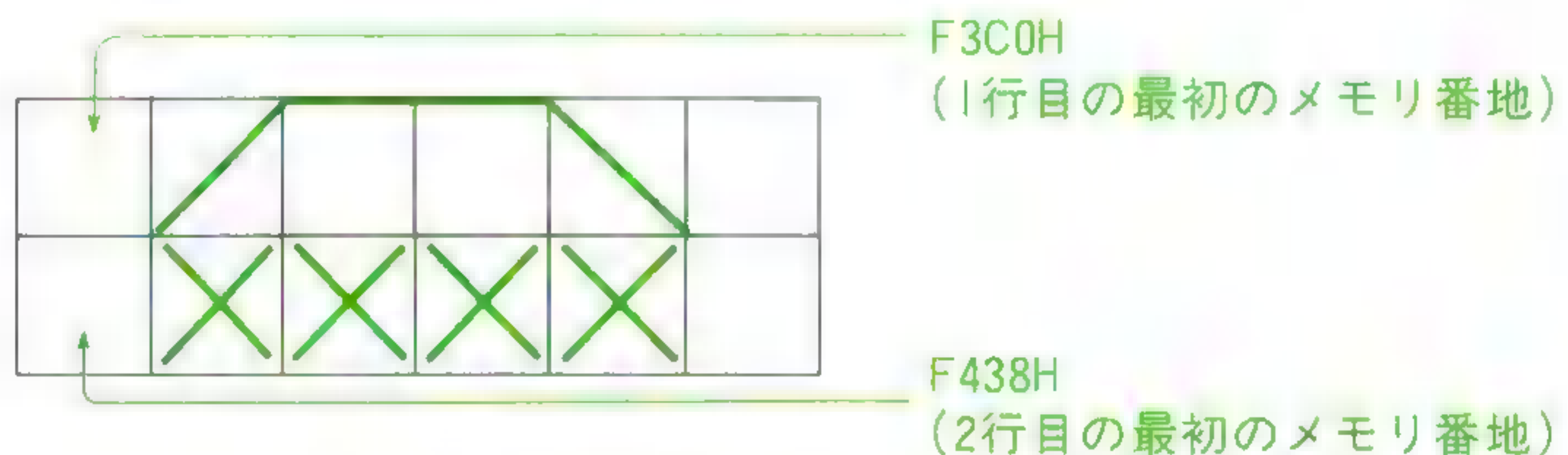
そして、ADD命令で、HLレジスタに取り出してきたF3C0Hと、DEレジスタの0078H（10進数で120）を足して、2行目の最初のVRAMのメモリ番地を求めます。

ADD HL, DE

HLレジスタの内容と、DEレジスタの内容を足して、

F3C0H+0078H=F438H

足し算の結果のF438Hが、2行目の最初のVRAMのメモリ番地です。



このF438Hは、HLレジスタに記憶されますから、DJNZ命令で指定したメモリ番地の命令にジャンプさせて繰り返えしますが、この場合は、まえの場合と違って、ひとつやることがあります。

POP BC命令で02Hを取り出す

DJNZ命令で指定したメモリ番地にジャンプするとき、繰り返えす回数を代入したBレジスタの内容から1を引いてジャンプします。この場合のDJNZ命令の繰り返えしの回数は、最初にBレジスタに代入した02Hです。この02Hは、HLレジスタにほかの値を代入するので、破壊されないためにスタックに退避させましたから、POP命令でスタックから取り出していきます。

POP BC

これで、Bレジスタに02Hが取り出されました。したがって、つぎにDJNZ命令を置いて、実行を戻します。

DJNZ <PUSH BC命令のメモリ番地>

この場合、DJNZ命令で戻すメモリ番地は、PUSH BC命令が記憶されているメモリ番地です。PUSH BC命令に戻すのは、Bレジスタの内容02Hから、DJNZ命令でジャンプするために01Hを引いた結果の01Hが、まだBレジスタに記憶されているからです。したがって、PUSH BC命令にジャンプさせて、この01Hをスタックに退避させるわけです。

DJNZ命令でPUSH BC命令に戻ってBレジスタの内容01Hをスタックに退避してからは、さきに説明したように、それぞれの命令を処理して、HLレジスタに記憶されている2行目のVRAMのメモリ番地F348Hから、00Hを6個表示します。そして2行目のキャラクタを消去します。

消去し終わると、POP BC命令で、スタックに退避した01Hを取り出して、DJNZ命令で、Bレジスタの内容01Hから1を引きます。つまり、

$$01H - 01H = 00H$$

を行うわけです。その結果は0ですから、DJNZ命令でジャンプすることはやめて、つぎの命令の実行に移ります。

アセンブルしたプログラム

では、つぎにアセンブルしたプログラムを示します。これまで説明してきたUFOを消去する命令は、UFOを移動し終わったあと実行すればよいので、UFOを移動させる命令のあとのメモリ番地、D03FHからD054Hに追加しています。なお、実行させるときは、画面を80×25モードにして実行してください。



メモリ番地	マシン語	アセンブリ言語
D000	06 01	LD B, 01H
D002	21 7A F3	LD HL, F37AH
D005	C5	PUSH BC
D006	E5	PUSH HL
D007	06 02	LD B, 02H
D009	11 60 D0	LD DE, D060H
D00C	C5	PUSH BC
D00D	E5	PUSH HL
D00E	0E 01	LD C, 01H
D010	1A	LD A, (DE)
D011	77	LD (HL), A
D012	23	INC HL
D013	13	INC DE
D014	0C	INC C
D015	79	LD A, C
D016	FE 07	CP 07H
D018	C2 10 D0	JP NZ D010H
D01B	E1	POP HL
D01C	01 78 00	LD BC, 0078H
D01F	09	ADD HL, BC
D020	C1	POP BC
D021	10 E9	DJNZ D00CH
D023	16 00	LD D, 00H
D025	1E 00	LD E, 00H
D027	1C	INC E
D028	7B	LD A, E
D029	FE 32	CP 32H
D02B	C2 27 D0	JP NZ D027H
D02E	14	INC D
D02F	7A	LD A, D
D030	FE 78	CP 78H

D032	C2 25 D0	JP NZ D025H
D035	E1	POP HL
D036	C1	POP BC
D037	04	INC B
D038	23	INC HL
D039	78	LD A, B
D03A	FE 48	CP 48H
D03C	C2 05 D0	JP NZ D005H
D03F	06 02	LD B, 02H
D041	21 C0 F3	LD HL, F3C0H
D044	11 78 00	LD DE, 0078H
D047	C5	PUSH BC
D048	E5	PUSH HL
D049	06 06	LD B, 06H
D04B	3E 00	LD A, 00H
D04D	77	LD (HL), A
D04E	23	INC HL
D04F	10 FC	DJNZ D04DH
D051	E1	POP HL
D052	19	ADD HL, DE
D053	C1	POP BC
D054	10 F1	DJNZ D047H
D056	C3 66 5C	JP 5C66H

UFDのデータ

メモリ番地は、消去の命令が追加されたので、D060Hからに変更します。

D060	00 EE 94
D063	94 EF 00
D066	00 F0 F0
D069	F0 F0 00

タイトルの表示

最後に、タイトルを表示します。ここでは、つぎの実行結果に示すように、「GAME・SET」と表示することにします。

GAME・SET

実行結果

500
150000

まず、表示させたいタイトルをグラフィック・シンボルで書きます。そしてそのグラフィック・シンボルを、キャラクタコードに直します。キャラクタコードは16進数で書き表わします。これがデータとなります。

「GAME・SET」を表示するデータをつぎに示します。

D030	E4 94 94 94 E5 00 E4 94
D038	94 94 E5 00 87 00 00 00
D040	87 00 E4 94 94 94 E6 00
D048	00 00 00 E4 94 94 94 E6
D050	00 E4 94 94 94 E6 00 E7
D058	94 87 94 E6 87 00 00 00
D060	87 00 87 00 00 00 87 00
D068	87 EF 80 EE 87 00 87 80
D070	80 00 00 00 00 00 00 87
D078	80 80 80 00 00 87 80 80
D080	00 00 00 00 00 87 00 00
D088	87 00 00 00 00 00 87 80

1行目を表示する
データ

2行目を表示する
データ

D090	80	80	87	00	87	00	00	00	
D098	87	00	87	00	00	00	00	00	← 3行目を表示するデータ
DOA0	87	87	00	00	00	00	00	87	
DOA8	00	87	00	00	00	00	00	00	
DOB0	00	87	00	00	87	00	E6	94	
DOB8	87	00	87	00	00	00	87	00	
DOC0	87	00	00	00	87	00	87	00	← 4行目を表示するデータ
DOC8	00	00	00	00	00	00	00	00	
DOD0	00	00	00	87	00	87	00	00	
DOD8	00	00	00	00	00	87	00	00	
DOE0	E6	80	80	80	E7	00	87	00	
DOE8	00	00	87	00	87	00	00	00	
DOF0	87	00	E6	80	80	80	E4	00	← 5行目を表示するデータ
DOF8	00	00	00	E5	80	80	80	E7	
D100	00	E6	80	80	80	E4	00	00	
D108	00	87	00	00	00	FF	FF	FF	

「GAME・SET」は、5行で表示されるようになっており、データの数は、各行とも44個です。

タイトルを表示するプログラムを、入力しやすいように、ダンプリストで示します。実行するときは、画面を80×25モードにして、このプログラムのあとに、先に示したデータを続けて入力してください。

D000	06	05	21	5C	F6	11	30	D0	ダンプリスト
D008	C5	E5	0E	01	1A	77	23	13	
D010	0C	79	FE	2D	C2	0C	D0	E1	
D018	01	78	00	09	C1	10	E9	C3	
D020	66	5C	00	00	00	00	00	00	

2進↔16進変換表

16 進 数	2 進 数
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

レジスタの種類

種 類	記号	名 称	ビット
専用レジスタ	PC	プログラム・カウンタ	16
	SP	スタック・ポインタ	16
	IX	インデックス・レジスタ	16
	IY	インデックス・レジスタ	16
	I	インタラプト・ページ・アドレス・レジスタ	8
	R	メモリ・リフレッシュ・レジスタ	8
アキュームレ ータとフラグ	A	アキューム・レータ	8
	F	フラグ	8
	A'	アキューム・レータ(サブ)	8
	F'	フラグ(サブ)	8
汎用レジスタ	B	メイン・レジスタ	8
	C		8
	D		8
	E		8
	H		8
	L		8
	B'	サブ・レジスタ	8
	C'		8
	D'		8
	E'		8
	H'		8
	L'		8

● Z-80 CPUには、いろいろなレジスタがあります。プログラムを作るとき、ベーシックの変数のようにデータを代入したりするレジスタは、アキュームレータとフラグの項にあるAレジスタと、汎用レジスタの項にあるB、C、D、E、H、Lなどの各レジスタです。

Aレジスタは、アキュームレータと呼ばれ、ALU（演算論理回路）と一緒に 演算時の数値や演算結果などを記憶したりするなど、特別な働きをします。

Z-80アセンブル表

8ビット

×	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	n	(nn)	I	R
LD A,×	7 F	7 8	7 9	7 A	7 B	7 C	7 D	7 E	0 A	1 A	DD 7 E d	FD 7 E d	3 E n	3 A n n	ED 5 7	ED 5 F
LD B,×	4 7	4 0	4 1	4 2	4 3	4 4	4 5	4 6			DD 4 6 d	FD 4 6 d	0 6 n			
LD C,×	4 F	4 8	4 9	4 A	4 B	4 C	4 D	4 E			DD 4 E d	FD 4 E d	0 E n			
LD D,×	5 7	5 0	5 1	5 2	5 3	5 4	5 5	5 6			DD 5 6 d	FD 5 6 d	1 6 n			
LD E,×	5 F	5 8	5 9	5 A	5 B	5 C	5 D	5 E			DD 5 E d	FD 5 E d	1 E n			
LD H,×	6 7	6 0	6 1	6 2	6 3	6 4	6 5	6 6			DD 6 6 d	FD 6 6 d	2 6 n			
LD L,×	6 F	6 8	6 9	6 A	6 B	6 C	6 D	6 E			DD 6 E d	FD 6 E d	2 E n			
LD(HL),×	7 7	7 0	7 1	7 2	7 3	7 4	7 5						3 6 n			
LD(BC),×	0 2															
LD(DE),×	1 2															
LD(IX+d),×	DD 7 7 d	DD 7 0 d	DD 7 1 d	DD 7 2 d	DD 7 3 d	DD 7 4 d	DD 7 5 d						DD 3 6 d n			
LD(IY+d),×	FD 7 7 d	FD 7 0 d	FD 7 1 d	FD 7 2 d	FD 7 3 d	FD 7 4 d	FD 7 5 d						FD 3 6 d n			
LD(nn),×	3 2 n n															
LD I,×	ED 4 7															
LD R,×	ED 4 F															
ADD A,×	8 7	8 0	8 1	8 2	8 3	8 4	8 5	8 6			DD 8 6 d	FD 8 6 d	C 6 n			
ADC A,×	8 F	8 8	8 9	8 A	8 B	8 C	8 D	8 E			DD 8 E d	FD 8 E d	C E n			
SUB ×	9 7	9 0	9 1	9 2	9 3	9 4	9 5	9 6			DD 9 6 d	FD 9 6 d	D 6 n			
SBC A,×	9 F	9 8	9 9	9 A	9 B	9 C	9 D	9 E			DD 9 E d	FD 9 E d	D E n			
AND ×	A 7	A 0	A 1	A 2	A 3	A 4	A 5	A 6			DD A 6 d	FD A 6 d	E 6 n			
XOR ×	A F	A 8	A 9	A A	A B	A C	A D	A E			DD A E d	FD A E d	E E n			
OR ×	B 7	B 0	B 1	B 2	B 3	B 4	B 5	B 6			DD B 6 d	FD B 6 d	F 6 n			
CP ×	B F	B 8	B 9	B A	B B	B C	B D	B E			DD B E d	FD B E d	F E n			
INC ×	3 C	0 4	0 C	1 4	1 C	2 4	2 C	3 4			DD 3 4 d	FD 3 4 d				
DEC ×	3 D	0 5	0 D	1 5	1 D	2 5	2 D	3 5			DD 3 5 d	FD 3 5 d				

Z-80アセンブル表

16ビット

×	BC	DE	HL	SP	IX	IY	AF	nn	(nn)
LD AF,×									
LD BC,×								0 1 nn	ED 4 nn B
LD DE,×								1 1 nn	ED 5 nn B
LD HL,×								2 1 nn	2 A nn
LD SP,×			F 9		DD F 9	FD F 9		3 1 nn	ED 7 nn B
LD IX,×								DD 2 nn	DD 2 nn A
LD IY,×								FD 2 nn	FD 2 nn A
LD(nn),×	ED 4 nn	ED 5 nn	2 2 nn	ED 7 nn	DD 2 nn	FD 2 nn			
PUSH ×	C 5	D 5	E 5		DD E 5	FD E 5	F 5		
POP ×	C 1	D 1	E 1		DD E 1	FD E 1	F 1		
ADD HL,×	0 9	1 9	2 9	3 9					
ADD IX,×	DD 0 9	DD 1 9		DD 3 9	DD 2 9				
ADD IY,×	FD 0 9	FD 1 9		FD 3 9		FD 2 9			
ADC HL,×	ED 4 A	ED 5 A	ED 6 A	ED 7 A					
SBC HL,×	ED 4 2	ED 5 2	ED 6 2	ED 7 2					
INC ×	0 3	1 3	2 3	3 3	DD 2 3	FD 2 3			
DEC ×	0 B	1 B	2 B	3 B	DD 2 B	FD 2 B			

CPUコントロール

HALT	76
------	----

ジャンプ, コール, リターン

×	UN COND	C	NC	Z	NZ	PE	PO	M	P	
J P ×, nn	C 3 nn	DA nn	D 2 nn	CA nn	C 2 nn	EA nn	E 2 nn	FA nn	F 2 nn	
JR ×, e	1 8 e-2	3 8 e-2	3 0 e-2	2 8 e-2	2 0 e-2					
JP(HL)	E 9									
JP(IX)	DD E 9									
JP(IY)	FD E 9									
CALL ×, nn	CD nn	DC nn	D 4 nn	CC nn	C 4 nn	EC nn	E 4 nn	FC nn	F 4 nn	
DJNZ e										1 0 e-2
RET ×	C 9	D 8	D 0	C 8	C 0	E 8	E 0	F 8	F 0	
RETI	ED 4 D									
RETN	ED 4 5									

入 力

IN A, (n)	DB n
IN A,(C)	ED 7 8
IN B,(C)	ED 4 0
IN C,(C)	ED 4 8
IN D,(C)	ED 5 0
IN E,(C)	ED 5 8
IN H,(C)	ED 6 0
IN L,(C)	ED 6 8

10進-16進変換表

下位 上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

最上位× 桁	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
×00	256	512	768	1024	1280	1536	1792	2048	2304	2560	2816	3072	3328	3584	3840
×000	4096	8192	12288	16384	20480	24576	28672	32768	36864	40960	45056	49152	53248	57344	61440
×0000	65536														

●10進数・16進数変換表を使って、FEHを10進数に変換するときは、上位桁Fと、下位桁Eの交わったところを見ます。FEHは254であることがわかります。10進数40を16進数に変換するときは、40のところから、上位桁、下位桁を見ます。上位桁は2、下位桁は8なので、28Hとなります。

下の表はFFH以上のとき使います。100Hのときは、×00の×に1をあてはめて、100Hとして見ます。1FFHのときは、100で見て、上の表のFFHを見、その値を合計します。

$$100H = 256 \quad FFH = 255 \quad 256 + 255 = 511$$

1バイト符号付16進数

下位 上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	-128	-127	126	-125	-124	-123	-122	-121	-120	-119	-118	-117	-116	-115	-114	-113
9	-112	-111	-110	-109	-108	-107	-106	-105	-104	-103	-102	-101	-100	-99	-98	-97
A	-96	-95	-94	-93	-92	-91	-90	-89	-88	-87	-86	-85	-84	-83	-82	-81
B	-80	-79	-78	-77	-76	-75	-74	-73	-72	-71	-70	-69	-68	-67	-66	-65
C	-64	-63	-62	-61	-60	-59	-58	-57	-56	-55	-54	-53	-52	-51	-50	-49
D	-48	-47	-46	-45	-44	-43	-42	-41	-40	-39	-38	-37	-36	-35	-34	-33
E	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17
F	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

VRAMメモリ番地表

(0～255)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	F3 00	02	04	06	08	0A	0C	0E	F3 10	12	14	16	18	1A	1C	1E	F3 20	22	24	26
1	F3 78	7A	7C	7E	F3 80	82	84	86	88	8A	8C	8E	F3 90	92	94	96	98	9A	9C	9E
2	F3 F0	F2	F4	F6	F8	FA	FC	FE	F4 00	02	04	06	08	0A	0C	0E	F4 10	12	14	16
3	F4 68	6A	6C	6E	F4 70	72	74	76	78	7A	7C	7E	F4 80	82	84	86	88	8A	8C	8E
4	F4 E0	E2	E4	E6	E8	EA	EC	EE	F4 F0	F2	F4	F6	F8	FA	FC	FE	F5 00	02	04	06
5	F5 58	5A	5C	5E	F5 60	62	64	66	68	6A	6C	6E	F5 70	72	74	76	78	7A	7C	7E
6	F5 D0	D2	D4	D6	D8	DA	DC	DE	F5 E0	E2	E4	E6	E8	EA	EC	EE	F5 F0	F2	F4	F6
7	F6 48	4A	4C	4E	F6 50	52	54	56	58	5A	5C	5E	F6 60	62	64	66	68	6A	6C	6E
8	F6 C0	C2	C4	C6	C8	CA	CC	CE	F6 D0	D2	D4	D6	D8	DA	DC	DE	F6 E0	E2	E4	E6
9	F7 38	3A	3C	3E	F7 40	42	44	46	48	4A	4C	4E	F7 50	52	54	56	58	5A	5C	5E
10	F7 B0	B2	B4	B6	B8	BA	BC	BE	F7 C0	C2	C4	C6	C8	CA	CC	CE	F7 D0	D2	D4	D6
11	F8 28	2A	2C	2E	F8 30	32	34	36	38	3A	3C	3E	F8 40	42	44	46	48	4A	4C	4E
12	F8 A0	A2	A4	A6	A8	AA	AC	AE	F8 B0	B2	B4	B6	B8	BA	BC	BE	F8 C0	C2	C4	C6
13	F9 18	1A	1C	1E	F9 20	22	24	26	28	2A	2C	2E	F9 30	32	34	36	38	3A	3C	3E
14	F9 90	92	94	96	98	9A	9C	9E	F9 A0	A2	A4	A6	A8	AA	AC	AE	F9 B0	B2	B4	B6
15	FA 08	0A	0C	0E	FA 10	12	14	16	18	1A	1C	1E	FA 20	22	24	26	28	2A	2C	2E
16	FA 80	82	84	86	88	8A	8C	8E	FA 90	92	94	96	98	9A	9C	9E	FA A0	A2	A4	A6
17	FA F8	FA	FC	FE	FB 00	02	04	06	08	0A	0C	0E	FB 10	12	14	16	18	1A	1C	1E
18	FB 70	72	74	76	78	7A	7C	7E	FB 80	82	84	86	88	8A	8C	8E	FB 90	92	94	96
19	FB E8	EA	EC	EE	FB F0	F2	F4	F6	F8	FA	EC	FE	FC 00	02	04	06	08	0A	0C	0E
20	FC 60	62	64	66	68	6A	6C	6E	FC 70	72	74	76	78	7A	7C	7E	FC 80	82	84	86
21	FC D8	DA	DC	DE	FC E0	E2	E4	E6	E8	EA	EC	EE	FC F0	F2	F4	F6	F8	FA	FC	FE
22	FD 50	52	54	56	58	5A	5C	5E	FD 60	62	64	66	68	6A	6C	6E	FD 70	72	74	76
23	FD C8	CA	CC	CE	FD D0	D2	D4	D6	D8	CA	CC	CE	FD E0	E2	E4	E6	E8	EA	EC	EE
24	FE 40	42	44	46	48	4A	4C	4E	FE 50	52	54	56	58	5A	5C	5E	FE 60	62	64	66

	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	28	2A	2C	2E	F3 30	32	34	36	38	3A	3C	3E	F3 40	42	44	46	48	4A	4C	4E
1	A0	A2	A4	A6	A8	AA	AC	AE	B0	B2	B4	B6	B8	BA	BC	BE	F3 C0	C2	C4	C6
2	18	1A	1C	1E	F4 20	22	24	26	28	2A	2C	2E	F4 30	32	34	36	38	3A	3C	3E
3	F4 90	92	94	96	98	9A	9C	9E	F4 A0	A2	A4	A6	A8	AA	AC	AE	F4 B0	B2	B4	B6
4	08	0A	0C	0E	F5 10	12	14	16	18	1A	1C	1E	F5 20	22	24	26	28	2A	2C	2E
5	F5 80	82	84	86	88	8A	8C	8E	F5 90	92	94	96	98	9A	9C	9E	F5 A0	A2	A4	A6
6	F8	FA	FC	FE	F6 00	02	04	06	08	0A	0C	0E	F6 10	12	14	16	18	1A	1C	1E
7	F6 70	72	74	76	78	7A	7C	7E	F6 80	82	84	86	88	8A	8C	8E	F6 90	92	94	96
8	E8	EA	EC	EE	F6 F0	F2	F4	F6	F8	FA	FC	FE	F7 00	02	04	06	08	0A	0C	0E
9	F7 60	62	64	66	68	6A	6C	6E	F7 70	72	74	76	78	7A	7C	7E	F7 80	82	84	86
10	D8	DA	DC	DE	F7 E0	E2	E4	E6	E8	EA	EC	EE	F7 F0	F2	F4	F6	F8	FA	FC	FE
11	F8 50	52	54	56	58	5A	5C	5E	F8 60	62	64	66	68	6A	6C	6E	F8 70	72	74	76
12	C8	CA	CC	CE	F8 D0	D2	D4	D6	D8	DA	DC	DE	F8 E0	E2	E4	E6	E8	EA	EC	EE
13	F9 40	42	44	46	48	4A	4C	4E	F9 50	52	54	56	58	5A	5C	5E	F9 60	62	64	66
14	B8	BA	BC	BE	F9 C0	C2	C4	C6	C8	CA	CC	CE	F9 D0	D2	D4	D6	D8	DA	DC	DE
15	FA 30	32	34	36	38	3A	3C	3E	FA 40	42	44	46	48	4A	4C	4E	FA 50	52	54	56
16	A8	AA	AC	AE	FA B0	B2	B4	B6	B8	BA	BC	BE	FA C0	C2	C4	C6	C8	CA	CC	CE
17	FB 20	22	24	26	28	2A	2C	2E	FB 30	32	34	36	38	3A	3C	3E	FB 40	42	44	46
18	98	9A	9C	9E	FB A0	A2	A4	A6	A8	AA	AC	AE	FB B0	B2	B4	B6	B8	BA	BC	BE
19	FC 10	12	14	16	18	1A	1C	1E	FC 20	22	24	26	28	2A	2C	2E	FC 30	32	34	36
20	88	8A	8C	8E	FC 90	92	94	96	98	9A	9C	9E	FC A0	A2	A4	A6	A8	AA	AC	AE
21	FD 00	02	04	06	08	0A	0C	0E	FD 10	12	14	16	18	1A	1C	1E	FD 20	22	24	26
22	78	7A	7C	7E	FD 80	82	84	86	88	8A	8C	8E	FD 90	92	94	96	98	9A	9C	9E
23	FD F0	F2	F4	F6	F8	FA	FC	FE	FE 00	02	04	06	08	0A	0C	0E	FE 10	12	14	16
24	68	6A	6C	6E	FE 70	72	74	76	78	7A	7C	7E	FE 80	82	84	86	88	8A	8C	8E

VRAMメモリ番地表 (80×25ワード)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	F3 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	F3 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	F3 20	21	22	23	24	25	26	27
1	F3 78	79	7A	7B	7C	7D	7E	7F	F3 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	F3 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
2	F3 F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F4 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	F4 10	11	12	13	14	15	16	17
3	F4 68	69	6A	6B	6C	6D	6E	6F	F4 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	F4 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
4	F4 E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F4 F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F5 00	01	02	03	04	05	06	07
5	F5 58	59	5A	5B	5C	5D	5E	5F	F5 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	F5 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
6	F5 D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	F5 E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F5 F0	F1	F2	F3	F4	F5	F6	F7
7	F6 48	49	4A	4B	4C	4D	4E	4F	F6 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	F6 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
8	F6 C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	F6 D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	F6 E0	E1	E2	E3	E4	E5	E6	E7
9	F7 38	39	3A	3B	3C	3D	3E	3F	F7 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	F7 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
10	F7 B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	F7 C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	F7 D0	D1	D2	D3	D4	D5	D6	D7
11	F8 28	29	2A	2B	2C	2D	2E	2F	F8 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	F8 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
12	F8 A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F8 B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	F8 C0	C1	C2	C3	C4	C5	C6	C7
13	F9 18	19	1A	1B	1C	1D	1E	1F	F9 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	F9 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
14	F9 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	F9 A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F9 B0	B1	B2	B3	B4	B5	B6	B7
15	FA 08	09	0A	0B	0C	0D	0E	0F	FA 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	FA 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
16	FA 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	FA 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	FA A0	A1	A2	A3	A4	A5	A6	A7
17	FA F8	F9	FA	FB	FC	FD	FE	FF	FB 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	FB 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
18	FB 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	FB 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	FB 90	91	92	93	94	95	96	97
19	FB E8	E9	EA	EB	EC	ED	EE	EF	FB F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	FC 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
20	FC 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	FC 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	FC 80	81	82	83	84	85	86	87
21	FC D8	D9	DA	DB	DC	DD	DE	DF	FC E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	FC F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
22	FD 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	FD 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	FD 70	71	72	73	74	75	76	77
23	FD C8	C9	CA	CB	CC	CD	CE	CF	FD D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	FD E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
24	FE 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	FE 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	FE 60	61	62	63	64	65	66	67

VRAMメモリ番地表 (80×25モード)

	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79			
0	28	29	2A	2B	2C	2D	2E	2F	F3	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	F3	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	
1	F3	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F3	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	F3	C0	C1	C2	C3	C4	C5	C6	C7
2	18	19	1A	1B	1C	1D	1E	1F	F4	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	F4	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	
3	F4	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	F4	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F4	B0	B1	B2	B3	B4	B5	B6	B7
4	08	09	0A	0B	0C	0D	0E	0F	F5	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	F5	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	
5	F5	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	F5	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	F5	A0	A1	A2	A3	A4	A5	A6	A7
6	F8	F9	FA	FB	FC	FD	FE	FF	F6	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	F6	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	
7	F6	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	F6	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	F6	90	91	92	93	94	95	96	97
8	E8	E9	EA	EB	EC	ED	EE	EF	F6	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F7	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
9	F7	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	F7	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	F7	80	81	82	83	84	85	86	87
10	D8	D9	DA	DB	DC	DD	DE	DF	F7	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F7	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	
11	F8	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	F8	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	F8	70	71	72	73	74	75	76	77
12	C8	C9	CA	CB	CC	CD	CE	CF	F8	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	F8	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	
13	F9	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	F9	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	F9	60	61	62	63	64	65	66	67
14	B8	B9	BA	BB	BC	BD	BE	BF	F9	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	F9	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	
15	FA	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	FA	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	FA	50	51	52	53	54	55	56	57
16	A8	A9	AA	AB	AC	AD	AE	AF	FA	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	FA	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	
17	FB	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	FB	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	FB	40	41	42	43	44	45	46	47
18	98	99	9A	9B	9C	9D	9E	9F	FB	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	FB	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	
19	FC	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	FC	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	FC	30	31	32	33	34	35	36	37
20	88	89	8A	8B	8C	8D	8E	8F	FC	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	FC	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	
21	FD	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	FD	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	FD	20	21	22	23	24	25	26	27
22	78	79	7A	7B	7C	7D	7E	7F	ED	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	FD	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	
23	FD	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	FE	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	FE	10	11	12	13	14	15	16	17
24	68	69	6A	6B	6C	6D	6E	6F	FE	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	FE	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	

キャラクタ・コード表

下位	上位															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D _E		0	@	P		p	—	⊥		—	タ	ミ	二	×
1	S _H	D ₁	:	1	A	Q	a	q	—	⊥	°	ア	チ	ム	⌈	円
2	S _X	D ₂		2	B	R	b	r	—	⊥	「	イ	ツ	メ	⊥	年
3	E _X	D ₃	#	3	C	S	c	s	—	⊥	」	ウ	テ	モ	⌋	月
4	E _T	D ₄	\$	4	D	T	d	t	—	—	,	エ	ト	ヤ	▲	日
5	E _Q	N _K	%	5	E	U	e	u	—	—	・	オ	ナ	ユ	▲	時
6	A _K	S _N	&	6	F	V	f	v	—		ヲ	カ	ニ	ヨ	▲	分
7	B _L	E _B	▼	7	G	W	g	w	—		ア	キ	ヌ	ラ	▲	秒
8	B _S	C _N	(8	H	X	h	x		「	イ	ク	ネ	リ	♠	
9	H _T	E _M)	9	I	Y	i	y		」		ケ	ノ	ル	♥	
A	L _F	S _B	*	:	J	Z	j	z		レ	エ	コ	ハ	レ	♦	
B	H _M	E _C	+	;	K	[k	{		」	オ	サ	ヒ	ロ	♣	
C	C _L	→	,	<	L	¥	l	:	—	「	ヤ	シ	フ	ワ	●	
D	C _R	←	—	=	M]	m	}	—	」	ユ	ス	ヘ	ン	○	
E	S _O	↑	・	>	N	^	n	~	—	」	ヨ	セ	ホ	“	/	
F	S _I	↓	/	?	O	—	o		+	ノ	ツ	ソ	マ	。	＼	

《本書を書いていた方々》

春田正夫／沢田 昭

小山郁夫／大沢昭二

田所信一

イラスト・波木博信

早わかりマシン語事典

1986年4月25日 発行 ©

執筆代表 春 田 正 夫

沢 田 昭

発 行 者 富 永 弘 一

印 刷 所 慶昌堂印刷株式会社

発行所 東京都台東区 株式 新星出版社
台東2丁目24 会社

電話 (831) 0743 郵便番号110 振替東京4-72233

ISBN4-405-06057-6

マイコン時代をリードする新星出版社の

わかりやすい即実戦・実用マイコンBooks

早わかり 関口 泰・山科敦之著 ● 880円

マイコン用語辞典

早わかり 田中一郎・小山郁夫著 ● 980円

ベーシック用語辞典

早わかり 大橋 均・田中一郎著 ● 1600円

ベーシック決まり文句

早わかり 大橋 均・田中一郎著 ● 1800円

ベーシック辞典

まんが きぎょうへい・さとう光著 ● 950円

パソコンゼミナール

まんが 田中一郎・愛沢ひろし著 ● 950円

パソコンの一般知識

まんが 北園一平・愛沢ひろし著 ● 950円

ベーシックプログラミング

絵でわかる 新井克彦・こしあきお著 ● 1000円

初歩のマイコン百科

山科敦之・吉田紀彦著 ● 680円

よくわかるマイコン入門

三宅 誠・佐藤清明著 ● 780円

初歩のマイコン入門

三木 守著 ● 950円

初歩のパソコン入門

PC-8801mkII 新家弘健著 ● 1500円

パソコングラフィックス

PC-9801 渋谷一男・新井克彦著 ● 1600円

図解16ビット&グラフィック

PC-6001 岡田慎一・冨塚啓二郎著 ● 1600円

わかるマシン語入門

Z80 若松登志樹著 ● 1200円

わかる機械語入門

MSX 大沢昭二・田中一郎著 ● 980円

はじめてのプログラミング

MSX 山下利秋著 ● 980円

はじめてのパソコン入門

ゲームで覚える 川中篤胤著 ● 980円

MSXベーシック

PC-8001 ランダム・プロ著 ● 1300円

ゲーム・ライブラリー

PC-8001 友部誠一・清水一夫著 ● 1300円

Nベーシックとゲームプログラミング

PC-8001mkII 友部誠一・清水一夫著 ● 1300円

ゲーム・プログラミング

PC-6001,8001 K S Sマイコンプロ著 ● 1300円

ゲーム・ライブラリー

PC-6001 K S Sマイコンプロ著 ● 1300円

ゲーム・ライブラリー

PC-8001 門井弘美・斎藤秋生著 ● 1300円

すぐに役立つビジネスプログラミング

PC-1500/1250,1251 新井克彦著 ● 1600円

よくわかるポケコン活用法

MZ-2000,2200 SMCマイコンプロ著 ● 1300円

ゲーム・ライブラリー

MZ-700 SMCマイコンプロ著 ● 1300円

ゲーム・プログラミング

MZ-700シリーズ 大沢昭二・田中一郎著 ● 1300円

ゲームプログラムレッスン

MZ-80シリーズ SMCマイコンプロ著 ● 1300円

ゲーム・プログラミング

MZ-80 SMCマイコンプロ著 ● 1300円

ゲーム・ライブラリ

★ 新星出版社

〒110 東京都台東区台東2丁目24
電話(03)831-0743 振替東京4-72233

早わかり

マシン語事典

定価1600円



新星出版社

ISBN4-405-06057-6 C2055 ¥1600E